Desenvolvimento de Groupware Componentizado com Base no Modelo 3C de Colaboração

Marco Aurélio Gerosa

Tese de Douturado

Marco Aurélio Gerosa

Desenvolvimento de Groupware Componentizado com Base no Modelo 3C de Colaboração

Tese de Doutorado

Tese apresentada ao Programa de Pós-Graduação em Informática da PUC-Rio como requisito parcial para obtenção do título de Doutor em Informática.

Orientador: Hugo Fuks

Rio de Janeiro, fevereiro de 2006

Marco Aurélio Gerosa

Desenvolvimento de Groupware Componentizado com Base no Modelo 3C de Colaboração

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Hugo Fuks
Orientador
Departamento de Informática – PUC-Rio

Prof. Carlos José Pereira de Lucena
Departamento de Informática – PUC-Rio

Prof. Rubens Nascimento Melo
Departamento de Informática – PUC-Rio

Prof. Alberto Barbosa Raposo
Departamento de Informática – PUC-Rio

Prof. Marcos Roberto da Silva Borges
Departamento de Ciência da Computação – UFRJ

Profa. Ana Carolina Brandão Salgado
Departamento de Informática – UFPE

Prof. Sérgio Crespo Coelho da Silva Pinto
Departamento de Informática – UNISINOS

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico

Rio de Janeiro, 16 de março de 2006

Marco Aurélio Gerosa

Graduou-se em Engenharia da Computação na UFES (Universidade Federal do Espírito Santo) em 1999. Obteve o grau de Mestre em Informática em 2002 pela PUC-Rio. Durante o doutorado atuou no Laboratório de Engenharia de Software da PUC-Rio onde participou do desenvolvimento do ambiente de aprendizagem colaborativa AulaNet. Participou como palestrante de dezenas de congressos e de 2000 a 2005 publicou 47 artigos em livros, revistas e congressos. Atuou também como revisor de artigos para congressos e periódicos. Atualmente é professor e coordenador do curso de Ciência da Computação do Centro Universitário Vila Velha (UVV) e pesquisador associado ao Laboratório de Engenharia de Software da PUC-Rio.

Ficha Catalográfica

Gerosa, Marco Aurélio

Desenvolvimento de groupware componentizado com base no modelo 3C de colaboração / Marco Aurélio Gerosa ; orientador: Hugo Fuks. – Rio de Janeiro : PUC-Rio, Departamento de Informática, 2006.

275 f. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Engenharia de Software. 3. Groupware. 4. Desenvolvimento baseado em componentes. 5. Ambiente AulaNet. 6. Engenharia de groupware. 7. Sistemas colaborativos. I. Fuks, Hugo. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

Agradecimentos

Ao meu orientador, professor Hugo Fuks, que além dos ensinamentos sobre como ser um pesquisador, ensinou-me a perceber o mundo de uma maneira mais clara e madura. É de inestimável valor o que aprendi com ele ao longo destes 6 anos de trabalho juntos.

Ao professor Carlos José Pereira de Lucena, coordenador do Laboratório de Engenharia de Software (LES), pelo ambiente, pela infra-estrutura e pelas contribuições que alteraram o rumo da pesquisa.

Aos professores Sérgio Crespo, Ana Carolina Salgado, Marcos Roberto da Silva Borges, Alberto Raposo e Rubens Nascimento Melo por sua participação na banca e por suas valiosas contribuições no refinamento do trabalho. Agradeço também ao professor Mário Monteiro, por suas palavras de sabedoria, incentivo e amizade.

Aos meus colegas do projeto AulaNet, do LES e da PUC-Rio pelo companheirismo e ajuda prestados. Em especial a Mariano Pimentel e Celso Gomes pelo feedback contínuo, principalmente nas fases preliminares do trabalho.

Ao CNPq e à Fundação Padre Leonel Franca pelo apoio financeiro dado durante a realização deste trabalho, que foi parcialmente financiado por meio do processo nº 140103/2002-3 e do projeto Sistemas Multi-Agentes para a Engenharia de Software (ESSMA) bolsa nº 552068/2002-0.

À minha família e amigos pelos momentos subtraídos de nossa convivência para serem investidos na pesquisa. Agradecimentos especiais aos meus pais Aurélio Gerosa e Virgínia de Almeida Gerosa pela atenção e investimentos na minha educação. Agradeço também pelo apoio e incentivo constantes, que mesmo a muitos quilômetros de distância, foram essenciais para o sucesso deste trabalho.

E a todos aqueles que direta ou indiretamente contribuíram para a realização deste trabalho.

Resumo

Gerosa, Marco Aurélio; Fuks, Hugo. **Desenvolvimento de Groupware Componentizado com Base no Modelo 3C de Colaboração**. Rio de Janeiro, 2006. 275p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Groupware é evolutivo e é difícil de construir e de manter, e acaba tendo um código desorganizado e difícil de evoluir. Nesta tese é proposta uma abordagem de desenvolvimento de groupware baseado em componentes concebidos em função do modelo 3C de colaboração. No modelo 3C, a colaboração é analisada a partir da comunicação, coordenação e cooperação. Na abordagem proposta, parte-se das necessidades de colaboração do grupo, analisadas com base no modelo, e componentes de software também organizados em função do modelo são utilizados para compor a solução. Como estudo de caso, a abordagem é aplicada no desenvolvimento da nova versão do ambiente AulaNet, cujo código padece dos problemas mencionados anteriormente. Neste estudo de caso, são desenvolvidos *component kits* para instanciar os serviços de comunicação do ambiente. Os componentes possibilitam compor, recompor e customizar os serviços de modo a refletir alterações na dinâmica de colaboração.

Palavras-chave

Sistemas colaborativos; desenvolvimento de groupware; modelo 3C de colaboração; componentes de software.

Abstract

Gerosa, Marco Aurélio; Fuks, Hugo (Advisor). **Component-Based Groupware Development Based on the 3C Collaboration Model.** Rio de Janeiro, 2006. 275p. D.Sc. Thesis – Computer Science Department, Pontifical Catholic University of Rio de Janeiro.

Groupware is evolutionary and difficult to develop and maintain. Thus, its code becomes unstructured and difficult to evolve. In this thesis, a groupware development approach based on components organized according to the 3C collaboration model is proposed. In this model, collaboration is analyzed based on communication, coordination and cooperation. Collaboration necessities of the group, analyzed based on the 3C model, are mapped to software components, also organized according to the model, in order to compose the system. The proposed approach is applied as a case study to the development of the new version of the AulaNet environment. The environment's code currently suffers the mentioned problems. In order to instantiate the communication services of the environment, for the case study, 3C based component kits were developed. The components allow the composition, re-composition and customization of the services in order to reflect collaboration dynamics changes.

Keywords

Collaborative systems; groupware development; 3C collaboration model; software components.

Sumário

Lista de Figuras

Lista de Tabelas

Abreviaturas

API – Application Program Interface
AWT – Abstract Windowing Toolkit
BRETAM – Breakthrough, Replication, Empiricism, Theory, Automation, Maturity
BSCW – Basic Support for Cooperative Work
CFF – Component Framework Framework
CLOS – Common Lisp Object System
CLX – Component Library for Cross Platform
CMS – Content Management System
COCA – Collaborative Objects Coordination Architecture
COM – Component Objetc Model
COPSE – Collaborative Project Support Environment
CORBA – Common Object Request Broker Architecture
CRIWG – International Workshop on Groupware
CSCA – Computer Supported Collaborative Argumentation
CSCW – Computer Supported Cooperative Work
CSCWiD – Conference on CSCW in Design
CVEs – Collaborative Virtual Environments
DACIA – Dynamic Adjustment of Component InterActions
DBC – Desenvolvimento Baseado em Componentes
DISCIPLINE – DIstributed System for Collaborative Information Processing and LEarning
DAO – Data Access Objects
DTO – Data Transfer Object
EBR – First Seminar on Advanced Research in Electronic Business
FAQ – Frequently Asked Question
FTP – File Transfer Protocol
GPL – GNU General Public License
HTML – Hyper Text Transfer Protocol
IBIS – Issue Based Information Systems
IDE – Integrated Development Environment
IDL – Interface Definition Language
IIOP – Internet Inter-ORB Protocol
IJCIS – International Journal of Cooperative Information Systems
JAI – Jornada de Atualização em Informática
JAMM – Java Applets Made Multiuser
JCP – Java Community Process
JSF – Java Server Faces
JSP – Java Server Pages
LES – Laboratório de Engenharia de Software

MoCA – Mobile Collaboration Architecture
MVC – Model, View, Controller
OLE – Object Linking and Embedding
OMG – Object Management Group
OO – Orientação a objetos
PAC – Presentation, Abstraction and Control)
PDA – Personal Digital Assistant
POJO – Plain Old Java Object
RAD – Rapid Application Development
RD – Requisito de Desenvolvedor
RIA – Rich Internet Application
RPCs – Remote Procedure Calls
RU – Requisito de Usuário
RUP – Rational Unified Process
SBIE – Simpósio Brasileiro de Informática na Educação
SDG – Single Display Groupware
SDK – Software Development Kit
SGBD – Sistema Gerenciador de Banco de Dados
SOAP – Simple Object Access Protocol
SWT – Standard Widget Toolkit
TIAE – Tecnologias de Informação Aplicadas à Educação
UML – Unified Modeling Language
VNC – Virtual Networking Computing
VRML – Virtual Reality Modeling Language
WCSCW – Workshop Brasileiro de Tecnologias para Colaboração
WDBC – Workshop de Desenvolvimento Baseado em Componentes
Webmedia – Simpósio Brasileiro de Sistemas Multimídia e Web
WYSIWIS – What You See Is What I See
WYSIWYG – What You See Is What You Get
XML – Extensible Markup Language
XOOPS – eXtensible Object Oriented Portal System
XSL – Extensible Style Language

1
Introdução

Douglas Engelbart [1968] apontou a relevância das aplicações desktop para automação de escritório, para hipertexto e para grupos. Hoje, as duas primeiras são largamente difundidas, utilizadas e comercialmente aceitas, enquanto a tecnologia para groupware ainda é encarada como instável e comercialmente arriscada, além de possuir poucos produtos [Greenberg, 2006]. Em grande parte das empresas, o suporte computacional à colaboração se limita a sistemas para troca de mensagens ou arquivamento de documentos.

A tecnologia de groupware ainda não atingiu seu potencial de utilização. Greenberg [2006] argumenta que a pesquisa em CSCW (*Computer Supported Cooperative Work*) está bastante avançada, porém falta um ferramental que simplifique a programação de aplicações colaborativas e promova o avanço na criatividade e o estabelecimento de uma massa crítica de utilização [Markus & Connolly, 1990]. As aplicações desktop foram impulsionadas pelo advento das interfaces gráficas com o usuário e dos toolkits de widgets, que possibilitaram programadores medianos construir aplicações arrastando e configurando componentes. O advento do HTML, com sua simplicidade e tolerância a erros, e dos editores WYSIWYG (*What You See Is What You Get*) possibilitou que o desenvolvimento de hipertexto seja ensinado até em escolas primárias. No desenvolvimento de groupware ainda são necessários programadores qualificados, aptos a lidar com protocolos, conexões, compartilhamento de recursos, concorrência de acesso, distribuição, renderização, gerenciamento de sessões, etc. Isto limita a quantidade de desenvolvedores atuando na área e desloca a criatividade e os esforços destes desenvolvedores para a criação de soluções para os problemas de natureza técnica de baixo nível, delegando a investigação da interação e o suporte à colaboração para segundo plano [Greenberg, 2006].

Estas dificuldades de desenvolvimento de groupware são vivenciadas no desenvolvimento e manutenção do ambiente AulaNet. O AulaNet é um

groupware, aplicado no ensino-aprendizagem pela web [Lucena & Fuks, 2000]. O AulaNet vem sendo desenvolvido desde 1997 pelo Laboratório de Engenharia de Software da Pontifícia Universidade Católica do Rio de Janeiro (LES/PUC-Rio) e distribuído pela empresa EduWeb, que customiza e presta serviços relacionados ao ambiente. O AulaNet é utilizado no Brasil na PUC-Rio, UFRJ, UFBA, UFMG, UFMT, UCP, Católica de Salvador, Faculdade Michelangelo (DF), Faculdades Integradas Boituva (SP), Universidade de Macapá entre outras, e fora do Brasil na Universidade Tecnológica do Panamá, Universidade de Aveiro, Instituto Politécnico de Gaya, Universidade da Madeira, Fraunhofer de Berlim, entre outras. Também é usado em instituições como Nextel, Rede Globo, Alpargatas, Ultragás, SEST/Senat, Profarma, Polícia Civil do Estado do Rio de Janeiro, Inmetro, entre outras.

O grupo de desenvolvedores do AulaNet da PUC-Rio é composto por alunos do doutorado, mestrado e graduação, que além de mantê-lo, utilizam-no em suas teses, dissertações e monografias, implementando e testando os conceitos de seus trabalhos. O ambiente cresceu por prototipação e suas funcionalidades foram implementadas evolutivamente. As constantes mudanças no suporte à colaboração e a evolução das tecnologias utilizadas tornaram o código da aplicação fortemente acoplado e com baixa coesão. Aspectos técnicos permeiam todo o código, ficando misturado ao suporte à colaboração, desviando o foco do desenvolvedor. Mudanças no ambiente têm reflexo em diversas partes do código e causam indesejados efeitos colaterais, dificultando a evolução do ambiente, a integração de novos membros à equipe de desenvolvimento e a integração com a empresa EduWeb.

Este cenário ilustra a necessidade de se prover um ferramental para apoiar o desenvolvedor de groupware, para que ele desenvolva um sistema extensível, mais propício a acompanhar a evolução do suporte à colaboração e das características das tarefas e dos grupos envolvidos. Um ferramental que encapsule as complexidades de baixo nível propicia a investigação da interação através da prototipação, além de possibilitar que um desenvolvedor não tão especializado adapte e reconfigure a ferramenta para suas necessidades específicas, o que é desejável, visto que não há como antever todas as demandas da colaboração [Pumareja et al., 2004].

1.1.
A Tese

A questão investigada nesta tese é de como apoiar desenvolvedores de groupware na construção de groupware extensível e propício a acompanhar a evolução do suporte à colaboração. A hipótese desta tese é que utilizar componentes concebidos em função do modelo 3C (comunicação, coordenação e cooperação) possibilita o desenvolvimento de groupware extensível, cuja composição é guiada pelas necessidades de colaboração.

Ao enxergar o problema sob a perspectiva do modelo 3C e utilizar a componentização organizada em função deste modelo, as alterações na colaboração são mapeadas ao suporte computacional, que é substituído ou acrescentado na medida da necessidade. Ao prover ao engenheiro de software uma infra-estrutura componentizada específica para o domínio de groupware, fundamentada em um modelo de colaboração, espera-se instrumentar a construção e a manutenção de sistemas colaborativos extensíveis e adaptáveis. Os componentes possibilitam lidar com o projeto da colaboração em um alto nível.

Cada grupo que utiliza um ambiente colaborativo tem necessidades específicas de colaboração, normalmente não necessitando de todas as ferramentas disponíveis. Ao conceber e desenvolver estas ferramentas na forma de componentes de software, instrumenta-se o desenvolvedor de modo que ele monte um groupware específico para as necessidades de colaboração do grupo. As ferramentas são selecionadas de um *component kit* organizado em função do modelo 3C para apoiar a dinâmica estabelecida. O desenvolvedor seleciona entre os componentes de mesmo propósito aqueles mais adequados à situação em questão.

Um serviço colaborativo normalmente possui suporte a funcionalidades referentes aos três Cs. Estas funcionalidades são recorrentes e relativamente autocontidas, como gerenciamento de sessão, permissão, avaliação, percepção, etc. Estas funcionalidades também são encapsuladas em componentes organizados em função do modelo 3C. O desenvolvedor de uma ferramenta colaborativa seleciona os componentes 3C que atendam às características do suporte à

colaboração referentes à utilização da ferramenta no apoio à atividade colaborativa. Estes componentes encapsulam as complexidades técnicas e o suporte à colaboração e favorecem a concentração dos esforços do desenvolvedor na composição de um groupware específico. Os componentes encapsulam implementações e regras de negócio sobre colaboração, providas por especialistas do domínio e obtidas por experimentação, e reusadas em diversas situações. Desta forma, a modelagem é instrumentada com base em modelos e elementos pré-definidos.

O problema que esta tese endereça, além de freqüentemente citado na literatura, está sendo vivenciado nos 8 anos de desenvolvimento do AulaNet. O AulaNet vem evoluindo por prototipação e sua implementação se tornou desestruturada, necessitando de uma custosa reestruturação. A abordagem proposta é aplicada no re-desenvolvimento do ambiente AulaNet, como um estudo de caso. A nova versão do AulaNet está sendo desenvolvida com a capacidade de recompor o ambiente, de reusar seus serviços em outras situações e reconfigurá-los para acompanhar a evolução dos processos de trabalho e das características do grupo. É utilizada uma arquitetura organizada em camadas, contendo *component frameworks* para lidar com os serviços e com os componentes de colaboração, provenientes de um *component kit*. O *component kit* é obtido a partir de uma engenharia do domínio, que visa o reuso e a interoperabilidade [Werner & Braga, 2005].

Além da utilização no desenvolvimento da nova versão do AulaNet, foram buscados indícios da aceitação acadêmica da abordagem e a arquitetura desenvolvida foi comparada com outras abordagens encontradas na literatura. A abordagem proposta nesta tese também foi utilizada por alunos da disciplina Engenharia de Groupware do Departamento de Informática da PUC-Rio. As principais contribuições desta tese é a proposição de uma abordagem para o desenvolvimento de groupware baseado em componentes organizados em função do modelo 3C de colaboração; a definição de uma arquitetura baseada em component frameworks e component kits; e a elaboração de um component kit para instanciar serviços colaborativos.

1.2. O Consórcio de Pesquisa

Para investigar o desenvolvimento de groupware e sua aplicação no desenvolvimento do AulaNet 3.0, nosso grupo de pesquisa Groupware@LES consorciou três trabalhos: Barreto [2006], em sua dissertação de mestrado, propõe a integração de *frameworks* na constituição da arquitetura técnica do AulaNet; Gerosa, em sua tese de doutorado, propõe a montagem de groupware a partir da agregação de serviços e componentes baseados no Modelo 3C de Colaboração; e Pimentel [2006], em sua tese de doutorado, propõe um processo de desenvolvimento de groupware usando o Modelo 3C de Colaboração em diferentes etapas do processo. Estes trabalhos consorciados reduzem a distância semântica entre a implementação e os conceitos do domínio referentes à colaboração, o que favorece a manutenção e a evolução do groupware. Com o objetivo de contextualizar os três trabalhos, esta seção é replicada na introdução da dissertação e das teses resultantes. As subseções seguintes resumem cada trabalho.

1.2.1. Agregando Frameworks de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet

No desenvolvimento de um groupware, o projetista se depara com desafios em diferentes níveis: entender do domínio e lidar com questões de infra-estrutura. O desenvolvimento de groupware é difícil devido ao seu caráter multidisciplinar e à heterogeneidade dos diversos grupos de trabalho. O desenvolvedor deveria se concentrar mais nos aspectos funcionais utilizando uma infra-estrutura que trate as questões técnicas.

Na dissertação de Barreto [2006], foi elaborada uma arquitetura técnica multicamadas que faz uso do padrão MVC (model-view-controller) [Fowler, 2002] e que integra frameworks de infra-estrutura [Fayad & Schmidt, 1997; Fayad et al. 1999a; Fayad et al., 1999b; Fayad & Johnson, 2000]. A abordagem multicamadas com o padrão MVC proporciona a separação entre a lógica da aplicação e a interface com o usuário, considerada uma boa prática de design de

software [Fowler, 2002]. Frameworks de infra-estrutura proporcionam uma maneira de lidar com as questões de baixo nível como persistências de dados, controle de transações, segurança, entre outros.

O diagrama esquematizado na Figura 1.1 mostra a arquitetura técnica proposta para o AulaNet 3.0. As setas indicam o fluxo de controle da aplicação, os retângulos representam classes, os círculos representam as interfaces, e as linhas pontilhadas representam a divisão entre as camadas. Esta arquitetura, baseada na Arquitetura de POJOs (*Plain Old Java Object*) descrita por Johnson [2002; 2004], é organizada nas seguintes camadas: apresentação, negócios e recursos.
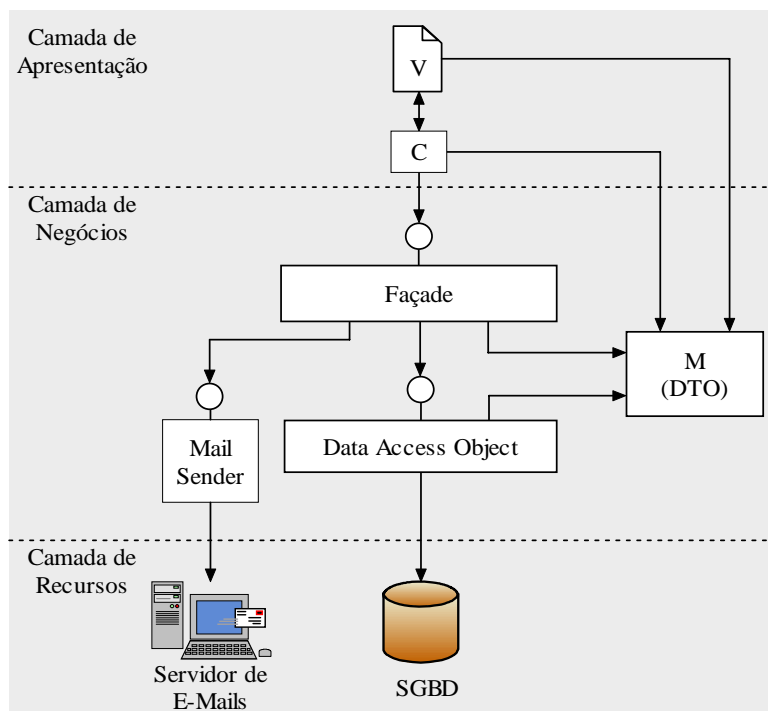
Figura 1.1. Arquitetura Técnica do AulaNet 3.0

A camada de recursos relaciona os recursos externos necessários para que a aplicação seja executada. Na arquitetura do AulaNet 3.0 estão previstos o uso de banco de dados relacional (SGBD) e um servidor de e-mails.

A camada de negócios implementa a lógica da aplicação utilizando POJOs [POJO, 2005].

> “O termo [POJO] foi cunhado enquanto eu [Martin Fowler], Rebbecca Parsons e Josh MacKenzie estávamos nos preparando para uma conferência em Setembro de 2000. Na palestra estávamos levantando os vários benefícios de codificar a lógica de negócios usando objetos Java comuns em vez de usar Beans de Entidade [EJB]. Questionávamos por que as pessoas eram tão contra usar objetos comuns em seus sistemas, e concluímos que era pela falta de um nome pomposo para os objetos simples. Então inventamos um, e o termo pegou muito bem.” [POJO, 2005]

Na camada de negócios, o modelo (representado no diagrama da Figura 1.1 pela letra M do MVC) é implementado por classes que realizam o padrão de projetos *Data Transfer Object* (DTO) [Fowler, 2002], usadas para transportar os dados das entidades de negócio entre camadas. O acesso à base de dados é encapsulado através de classes que implementam o padrão de projetos *Data Access Objects* (DAO) [Alur et al., 2001], o que possibilita variar a maneira de persistir as classes do modelo sem que seja preciso reescrever o código cliente. Mail Sender é a classe para enviar e-mails que, de forma similar ao DAO, encapsula o acesso ao servidor de e-mails. A lógica de negócios é exposta para a camada de apresentação através de um *Façade* [Gamma et al., 1995], que é o padrão de projeto para prover uma interface para acesso às funcionalidades de um serviço.

A camada de apresentação expõe a lógica de negócios ao usuário-final. Na arquitetura do AulaNet 3.0, esta camada é composta pelo controlador (representado no diagrama da Figura 1.1 pela letra C do MVC) e por páginas JSP que implementam a camada de Visão (representado no diagrama da Figura 1.1 pela letra V do MVC). O controlador chama os métodos do *Façade*, acessando a camada de negócios. Os DTOs resultantes de operações são passados à visão que exibe as informações ao usuário.

Frameworks de infra-estrutura foram selecionados e acrescentados a esta arquitetura para prover persistência de dados, gerenciamento de transações entre outros aspectos. Estes frameworks possibilitam ao desenvolvedor tratar estes aspectos com uma visão em alto nível, concentrando-se em seu domínio de aplicação, no caso, groupware.

1.2.2. Desenvolvimento de Groupware Componentizado com base no Modelo 3C de Colaboração

Um groupware é composto de ferramentas colaborativas como Fórum, Agenda, Documentação, etc. Estas ferramentas, disponíveis em diversas aplicações groupware, compartilham funcionalidades relativas ao suporte computacional à colaboração, tais como canal de comunicação, gerenciamento de participantes e registro de informações.

Nesta tese, para dar suporte ao desenvolvimento de groupware, foram estabelecidos dois níveis de componentização. O primeiro nível é constituído de serviços colaborativos que, por sua vez, são montados com componentes 3C (segundo nível) que implementam funcionalidades relacionadas à colaboração. Estes componentes são distribuídos em *component kits* organizados em função do modelo 3C de colaboração para que desenvolvedores montem aplicações colaborativas. Nesta abordagem, cada serviço usa componentes de comunicação, coordenação e de cooperação independentemente da classificação 3C do serviço. Foi aplicado um método de Engenharia do Domínio para elaborar o conjunto de componentes. Os componentes são iterativamente refinados em função da realimentação obtida com o desenvolvimento dos serviços do AulaNet 3.0 e em função de estudos de caso variando as configurações do suporte à colaboração.

Component frameworks [Syzperski, 1997] são usados para oferecer suporte ao gerenciamento e à execução dos componentes. Conforme apresentado na Figura 4.10, nesta tese foi elaborado um *component framework* para cada nível de componentização (serviço e componente 3C). Os serviços são acoplados no Service Component Framework, e os componentes 3C são acoplados no Collaboration Component Framework. Estes *component frameworks* são responsáveis por tratar a instalação, remoção, atualização, ativação, desativação, localização, configuração e monitoramento de componentes. O Service Component Framework gerencia as instâncias dos serviços e a ligação com os componentes de colaboração correspondentes. O Collaboration Component Framework gerencia as instâncias dos componentes de colaboração, que são provenientes do Collaboration Component Kit. Algumas funcionalidades dos *component frameworks* são recorrentes, sendo então elaborado um *framework* para instanciar os *component frameworks*. Este tipo de *framework* é chamado de *component framework framework* (CFF) [Szyperski, 1997, p.277]. Um *component framework framework* é visto como um *component framework* de segunda ordem, onde seus componentes são *component frameworks* [Szyperski, 1997, p.276]. Na arquitetura da aplicação, o *component framework* de segunda ordem foi denominado Groupware Component Framework Framework.
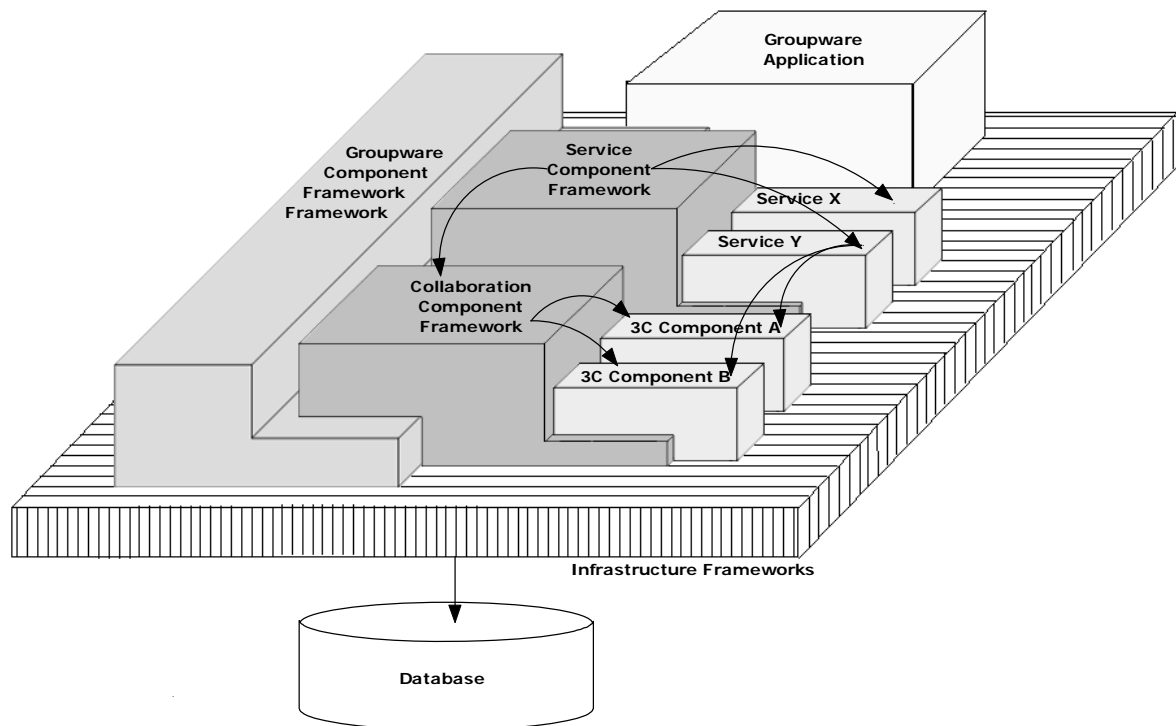
Figura 1.2. A arquitetura de aplicação proposta

Os *component frameworks*, serviços e componentes 3C oferecem suporte computacional aos conceitos do modelo 3C de colaboração, instrumentando o desenvolvimento da camada de negócio. A arquitetura de aplicação proposta estrutura os componentes do domínio, representando um projeto lógico de alto nível independente da tecnologia de suporte [D'Souza & Wills, 1998]. os aspectos de infra-estrutura, tratados na dissertação de Barreto [2006], são independentes do domínio de aplicação.

Os componentes da arquitetura de aplicação são implementados segundo a arquitetura técnica. Os serviços do AulaNet são criados com um único *Façade* que expõe as operações deste serviço para a camada de apresentação. Os componentes de colaboração por sua vez, podem utilizar vários DTOs e DAOs, dependendo da complexidade do componente. Estes componentes podem ainda usar "código cola" [Szyperski, 1997] e adaptadores [D'Souza & Wills, 1998] para possibilitar a integração com componentes e outros sistemas que não são compatíveis por construção.

1.2.3.
RUP-3C-Groupware: um Processo de Desenvolvimento de Groupware baseado no Modelo 3C de Colaboração

Os *frameworks* de infra-estrutura selecionados por Barreto [2006] se encarregam de soluções para aspectos de infra-estrutura de baixo nível, visando possibilitar o desenvolvedor se concentrar nos aspectos funcionais. Os *kits* de serviços e componentes 3C elaborados nesta tese fornecem os elementos para compor um groupware. O processo elaborado na tese de Pimentel [2006] estabelece os passos a serem seguidos na montagem do groupware, pois ainda que se construa uma aplicação groupware para um grupo com uma determinada dinâmica, com o tempo surgem novas situações onde são identificados novos problemas. A aplicação necessitará ser modificada para não se manter inadequada.

Um processo organiza, em linhas gerais, uma seqüência de passos onde são incorporadas diretrizes e boas práticas que, quando seguidas, levam à produção de um software [Sommerville, 2003; Beck, 2004; Philippe, 2003]. Coexistem abordagens diferentes para o desenvolvimento de software, dentre elas, o desenvolvimento baseado em componentes, que é uma estratégia recente que tem se tornado cada vez mais usada [Sommerville, 2003; Gimenes & Huzita, 2005]. Seguindo esta abordagem, tornaram-se conhecidos processos como Catalysis [D'Souza e Wills, 1998], *UML Components* [Cheesman & Daniels, 2001] e RUP – Rational Unified Process [Philippe, 2003]. O processo formalizado na tese de Pimentel [2006], denominado RUP-3C-Groupware, também faz uso da abordagem baseada em componentes, estendendo o RUP para o desenvolvimento específico de aplicações groupware.
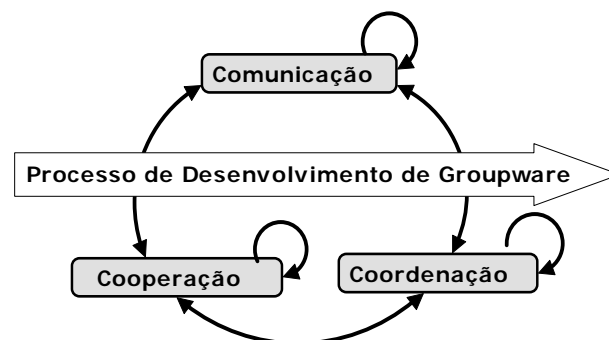
Figura 1.3. Foco para o desenvolvimento de uma versão da aplicação groupware com base no Modelo 3C de Colaboração

No processo proposto, o Modelo 3C de Colaboração é usado nas diferentes etapas do processo: na análise de domínio para classificação das aplicações groupware e de seus elementos; na construção de componentes; e no foco dado para o desenvolvimento de cada versão. De acordo com essa prática, a aplicação groupware é desenvolvida resolvendo um problema de comunicação, de coordenação ou de cooperação, um a cada versão ao longo do ciclo de desenvolvimento, esquematizado na Figura 1.3.

1.2.4.
Estrutura da Tese

A estrutura desta tese é a seguinte: no Capítulo 2 são discutidos alguns trabalhos relacionados ao desenvolvimento de groupware componentizado. O Capítulo 3 apresenta o modelo 3C de colaboração, utilizando o AulaNet e um de seus cursos como estudo de caso. No Capítulo 4, a abordagem utilizada e o ferramental desenvolvido são apresentados. No Capítulo 5 são apresentadas as instanciações, re-configurações e reuso dos serviços e seus componentes. O Capítulo 6 apresenta a conclusão e as direções futuras de pesquisa. O Apêndice A apresenta conceitos, modelos e tecnologias do desenvolvimento baseado em componentes. O Apêndice B apresenta a descrição dos componentes utilizados.

2
Revisão da Literatura

Neste capítulo é apresentada uma revisão de trabalhos relacionados. Na Seção 2.1, são apresentadas abordagens que estendem tecnologias desenvolvidas para a Engenharia de Software de uma forma específica para o desenvolvimento de groupware. Uma ênfase maior é dada às abordagens voltadas para o desenvolvimento de groupware baseado em componentes, dada a proximidade à proposta desta tese.

Neste capítulo também é feita uma revisão sobre a tecnologia de componentes de software (Seção 2.2) e de engenharia de domínio (Seção 2.4), que são subsídios para os demais capítulos da tese. Na Seção 2.2, são apresentados alguns termos e conceitos do desenvolvimento baseado em componentes, resumindo o Apêndice A, que detalha a componentização de software e o uso de frameworks. Optou-se por fazer uma revisão mais extensa na forma de apêndice, dada a variedade de definições e conceitos presentes na literatura da área.

Por fim, na Seção 2.3, a abordagem proposta nesta tese é comparada com algumas plataformas de desenvolvimento de groupware componentizado encontradas na literatura.

2.1.
Abordagens para o Desenvolvimento de Groupware

Vários trabalhos na literatura estendem as abordagens, técnicas, tecnologias e ferramentas da Engenharia de Software para o desenvolvimento de groupware, de modo a torná-las mais propícias para as características deste tipo de aplicação. Sendo a abordagem proposta nesta tese voltada para o desenvolvimento de groupware e estando esta tese no contexto de um projeto de pesquisa que visa instrumentar todo o ciclo de desenvolvimento de groupware, são apresentados nesta seção alguns exemplos de abordagens. São apresentados requisitos, UML estendida, padrões de projetos, arquiteturas de groupware, frameworks,

componentes, toolkits e técnicas de avaliação heurística específicas para o desenvolvimento de groupware.

2.1.1. Requisitos de Groupware

Groupware em geral apresenta requisitos recorrentes. Diversos trabalhos na literatura [Tietze, 2001; Schmidt & Rodden, 1996; Mandviwalla & Olfman, 1994] catalogam requisitos para instrumentar a especificação, análise e avaliação de groupware. Apesar de não ser possível especificar completamente um groupware a partir de uma lista de requisitos, as listas auxiliam a iniciar o processo, instrumentando o desenvolvedor. Nesta seção, são apresentados os requisitos propostos por Tietze [2001], por serem voltados para groupware baseado em componentes. Estes requisitos são utilizados no Capítulo 5 para analisar a arquitetura proposta.

Tietze [2001] divide seu catálogo de requisitos em dois grupos: *requisitos de usuário* (RU) e *requisitos de desenvolvedor* (RD). Os requisitos de usuário impactam diretamente os usuários finais do sistema colaborativo. Os requisitos de desenvolvedor impactam os desenvolvedores que utilizam a arquitetura de componentes para criar e adaptar as ferramentas colaborativas. Tietze propõe ao todo onze requisitos de usuário e nove de desenvolvedor. Tietze apresenta os requisitos através de cenários envolvendo personagens fictícios, que são sucintamente reproduzidos a seguir.

RU1 – Acesso aos objetos compartilhados e às ferramentas de colaboração – Andrew ao chegar para trabalhar em sua estação de trabalho deseja que o ambiente ofereça acesso aos documentos compartilhados e às ferramentas colaborativas para manipular os objetos e interagir com os demais participantes, oferecendo persistência das alterações promovidas nos objetos.

RU2 – Auxílio na escolha das ferramentas apropriadas – Andrew necessita escrever um relatório técnico cooperativamente com Bárbara, localizada remotamente. Dada a variedade de tarefas e ferramentas, o ambiente auxilia na escolha da ferramenta apropriada para a realização das tarefas e para a edição dos objetos compartilhados.

RU3 – Fornecimento de informações de percepção – Ao entrar no ambiente, Bárbara visualiza que Andrew está conectado e que um determinado capítulo do relatório está sendo editado por ele. Andrew também é notificado da disponibilidade de Bárbara.

RU4 – Colaboração síncrona e assíncrona – Andrew necessita discutir uma questão com Bárbara. O ambiente fornece serviços de colaboração síncrona e assíncrona, de modo que os participantes selecionam o modo de interação mais adequado a cada situação.

RU5 – Acesso ao ambiente independente da estação de trabalho – Bárbara necessita de informações presentes no laboratório da empresa. Ela deixa sua estação de trabalho e se conecta do computador do laboratório e continua a colaboração com Andrew do ponto onde foi interrompida.

RU6 – Fornecimento de espaço privativo e público e transição entre eles – Bárbara quer rascunhar algumas observações sem que Andrew tenha acesso. Quando estiverem mais consolidadas, ela convidará Andrew para visualizar e trabalhar no texto.

RU7 – Extensão dinâmica do ambiente – Andrew e Bárbara necessitam construir um determinado tipo de figura. Andrew busca na Internet uma ferramenta para este propósito e a incorpora ao ambiente, de modo que Bárbara também tenha acesso.

RU8 – Sincronização entre ferramentas diferentes – Bárbara abre uma ferramenta para manipular a estrutura do documento, enquanto Andrew utiliza o editor de texto. As alterações feitas em uma ferramenta são refletidas na outra.

RU9 – Mobilidade – Charles, o gerente do projeto, ao sair de uma reunião com um cliente, acessa o sistema através de seu PDA para verificar o andamento e as pendências.

RU10 – Agrupamento de ferramentas – Ao encerrar a sessão de colaboração, Andrew e Bárbara agrupam as ferramentas para possibilitar restaurar o ambiente mais rapidamente na próxima sessão.

RU11 – Alta performance – Ao colaborar, os participantes esperam que seu trabalho ocorra sem atrasos devido à latência do sistema De modo geral, as

ferramentas de colaboração síncronas têm requisitos mais fortes de *feedthrough* (modificações devido a ações dos companheiros) do que as ferramentas assíncronas, e operações interativas têm requisitos mais fortes de *feedback* (modificações devido a ações do indivíduo) do que operações em lote.

RD1 – Reuso da experiência e conhecimento anteriores – David, novo membro da equipe de desenvolvimento de ferramentas, deseja que sua experiência e conhecimento sobre programação de aplicações mono-usuário sejam aproveitados.

RD2 – Aproveitamento do modelo de dados – Visando a interoperabilidade e o reuso, David aproveita modelos de dados já existentes.

RD3 – Compartilhamento transparente de dados – David não deseja preocupar-se com a distribuição e com o compartilhamento de dados pelo sistema. A infra-estrutura provê recursos para gerenciar o acesso, a alocação e o compartilhamento dos dados.

RD4 – Suporte a dados locais e compartilhados – Nem todos os dados devem ser compartilhados. David decide se o dado vai ser local ou compartilhado, e a infra-estrutura provê recursos para tratar ambos da mesma maneira.

RD5 – Acesso às informações de percepção – David necessita acesso às informações necessárias para prover aos participantes informações de percepção.

RD6 – Disponibilização de novas ferramentas – A infra-estrutura não deve oferecer dificuldades para a disponibilização de novas ferramentas.

RD7 – Escalabilidade – A performance da infra-estrutura não se degrada perceptivelmente quando novos usuários se conectam.

RD8 – Integração com ferramentas externas – A infra-estrutura é adaptável para possibilitar a integração do ambiente a ferramentas externas.

RD9 – Suporte a ferramentas localizadas no servidor – A infra-estrutura provê recursos para que ferramentas sejam executadas no servidor, como nos casos em que é necessário acesso direto a determinados recursos ou execução ininterrupta.

2.1.2. UML Estendida

Ao projetar e desenvolver software, é necessária uma notação comum entre os envolvidos para que eles possam documentar e debater sobre o projeto do sistema. A notação de projeto que se tornou padrão e largamente utilizada pela indústria é a UML (*Unified Modeling Language*). A UML é uma notação extensível que inclui definições de diagramas de modelagem para as diversas atividades do desenvolvimento, desde as primeiras até as mais refinadas [Booch et al., 2000]. Para propósitos de extensibilidade, a UML prevê o uso de estereótipos, que são pontos de extensão dos diagramas para serem utilizados com particularidades do domínio em questão. Estereótipos são representados nos diagramas através de um nome precedido por “<<” e sucedido por “>>”.

Alguns trabalhos estendem a linguagem UML para a modelagem de aspectos específicos de groupware baseado em componentes. Estas extensões tornam mais direta a integração do projeto com a arquitetura utilizada. Na proposta de Rubart & Dawabi [2002], para representar aspectos específicos da arquitetura, as classes que representam objetos compartilhados são marcadas com o estereótipo <<shared>> e as que representam componentes de groupware, com o estereótipo <<component>>, conforme ilustrado na Figura 2.1. Estas extensões indicam implicitamente os mecanismos de encaixe na arquitetura, de concorrência, de sincronização, etc. Os componentes de groupware implementam as ferramentas colaborativas.
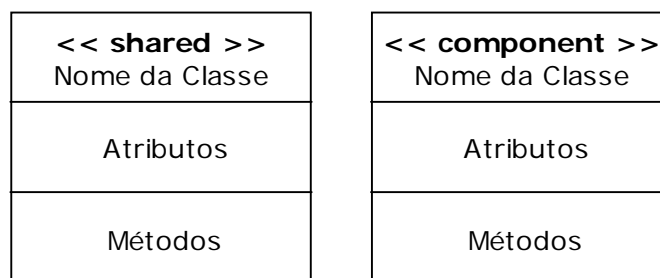
Figura 2.1. UML estendida apresentando classes que representam objetos compartilhados (à esquerda) e componentes de groupware (à direita) [Rubart & Dawabi, 2002]

A associação entre os componentes e os objetos compartilhados é realizada através de tarefas, o que leva à necessidade de representá-las no diagrama de classes. Uma tarefa é indicada através do estereótipo <<task>> nas relações entre as classes. Na Figura 2.2, dois componentes (um editor de documento e um

visualizador de estrutura) atuam em um mesmo objeto compartilhado (documento). As associações são realizadas através das tarefas editar e navegar. Na figura, um objeto da classe Documento contém outros objetos da mesma classe (um documento composto de seções). Como o objeto Documento é um objeto compartilhado, é representada a extensão <<slot>> antes dos nomes dos atributos. Este estereótipo faz a distinção entre os atributos regulares e os objetos compartilhados, que têm necessidades específicas de sincronização e integridade [Rubart & Dawabi, 2002].
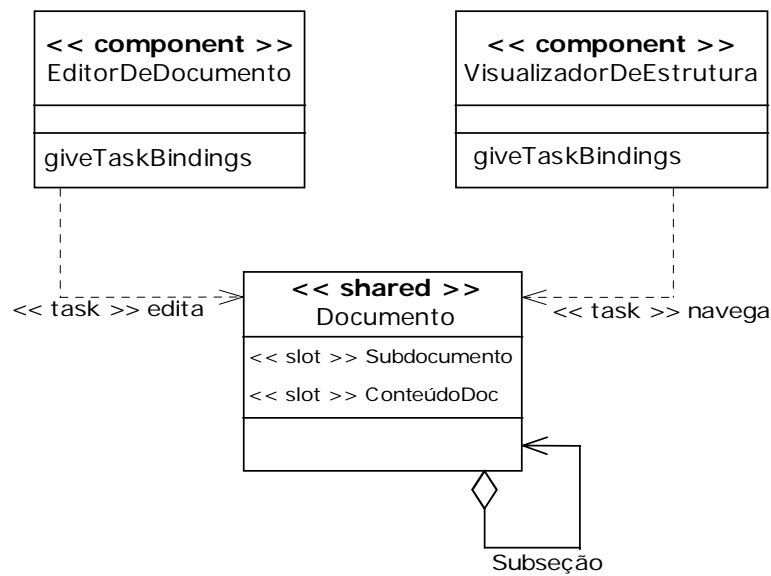
Figura 2.2. Diagrama de classes apresentando componentes de groupware que executam tarefas em um mesmo objeto compartilhado

As operações nos objetos compartilhados são realizadas durante uma sessão. Uma sessão é individual ou coletiva, sendo que a primeira pode ser transformada na segunda. Na Figura 2.3, são apresentadas duas sessões, que são representadas através do símbolo de componentes da linguagem UML marcados com a extensão <<session>>.
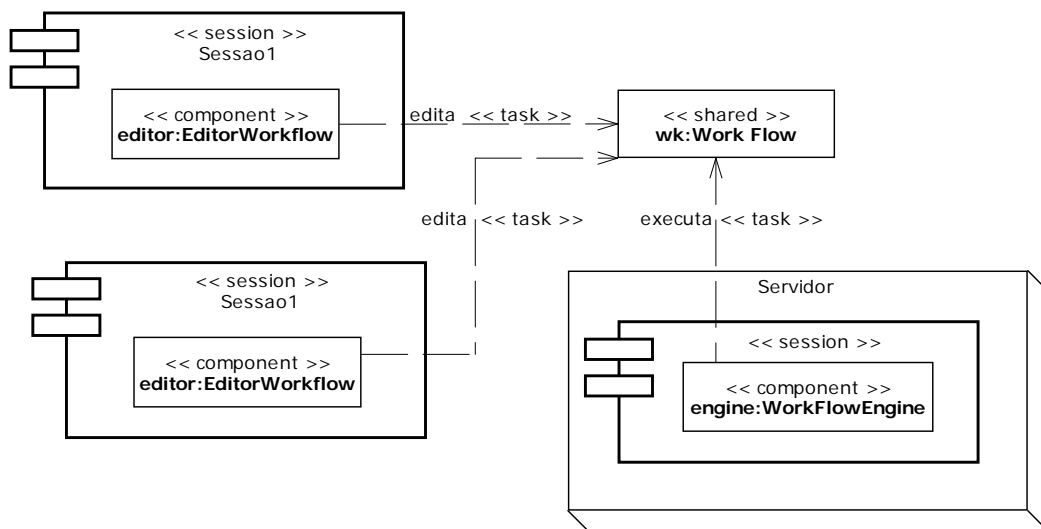
Figura 2.3. Extensão da UML para modelar sessões e nós de processamento

Para representar as operações realizadas no servidor, como por exemplo, a execução de uma instância de um fluxo de trabalho por uma máquina de workflow, utiliza-se uma sessão localizada em um nó representando o servidor, conforme pode ser observado na Figura 2.3. Quando o nó não for representado, assume-se uma estação cliente.

2.1.3. Padrões Específicos

Um padrão descreve um problema que foi identificado, definido, resolvido e documentado, direcionando a comunicação entre os desenvolvedores, o entendimento do sistema e o reconhecimento de problemas recorrentes. Os padrões catalogam problemas que os desenvolvedores enfrentaram, bem como as soluções dadas [Paludo & Burnett, 2005]. Com os padrões, a experiência dos desenvolvedores de software é documentada e reusada, registrando soluções, idéias e conceitos para um determinado problema ou contexto particular [Gamma et al., 1994]. Padrões de projeto registram uma solução recorrente ligada ao projeto do sistema, constituem uma linguagem comum entre os desenvolvedores e normalmente possuem suporte computacional implementado. Padrões de análise possuem objetivos similares aos padrões de projeto, porém com ênfase para as fases iniciais do ciclo de desenvolvimento de software. Um padrão de análise representa uma idéia que já foi útil em um contexto particular e que provavelmente será útil em outros [Fowler, 1996]. Os padrões de análise são

conceituais, com o objetivo de representar a forma de pensar do usuário em detrimento da forma utilizada pelo computador.

Alguns trabalhos propõem padrões de projetos específicos para instrumentar o desenvolvimento de groupware. Na Figura 2.4 são apresentadas famílias de padrões específicos para groupware, disponíveis em [Groupware Patterns Swiki, 2005]. Para cada padrão, é descrito intenção, família, nomes equivalentes, problema que endereça, cenário de uso, contexto, indicações, solução, participantes, racional da solução, usos conhecidos, padrões relacionados, referências bibliográficas e maneira de citar.
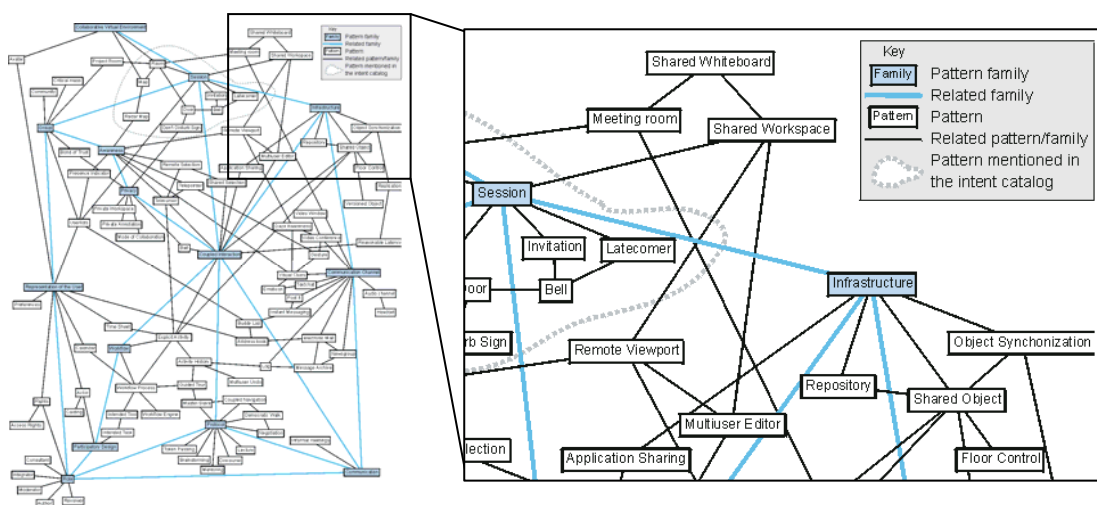
Figura 2.4. Padrões de projeto para groupware

Lukosch & Schümmer [2004] propõem uma linguagem de padrões para groupware, visando instrumentar o desenvolvimento, de modo a favorecer o reuso de soluções e capacitar mais rapidamente membros inexperientes no desenvolvimento de groupware. Também é considerada a integração com frameworks existentes, de modo a valorizar o reuso de decisões de projeto bem sucedidas e de conhecimento provido por especialistas.

Santoro et al. [2001] propõem um modelo baseado em padrões conceituais que descrevem situações e problemas comuns em um ambiente de ensino-aprendizagem e apresentam soluções recorrentes e reusáveis. Os padrões objetivam instrumentar o desenvolvedor de um ambiente no levantamento de necessidades, de modo a não deixar de fora questões importantes. Alguns dos padrões são transformados em padrões de projeto e outros servem de guia para o desenvolvimento. O modelo é apresentado em mais detalhes na Seção 3.8 do capítulo seguinte.

Kolfschoten et al. [2004] propõem uma engenharia de colaboração baseada em thinkLets, que são padrões de intervenção e concepção da dinâmica de colaboração do grupo visando incentivar um determinado padrão de colaboração. Os autores classificam os thinkLets de acordo com seu propósito principal para a fase da colaboração.

2.1.4. Arquiteturas de Groupware

Alguns trabalhos propõem arquiteturas específicas para groupware. Normalmente a arquitetura de um groupware provê mecanismos de controle de acesso, de compartilhamento, de concorrência e de sincronização entre objetos. Na Figura 2.5 encontra-se a arquitetura para groupware baseado em componentes proposta por Tietze [2001]. Cada participante utiliza uma aplicação cliente que contém os componentes de groupware. Estes componentes são instanciados a partir do repositório do *Component Broker*, localizado no servidor. Cada componente combina uma camada de interface, onde são implementadas as funcionalidades relativas à interação com o usuário, e uma camada de lógica de aplicação, onde se localizam as funcionalidades específicas do componente e da manipulação dos objetos. Os componentes acessam os objetos através do Modelo de Dados do Domínio, que implementa as funcionalidades relativas ao domínio em questão e é responsável por manter as consistências semânticas entre os objetos. O Modelo de Dados copia os objetos compartilhados localizados no Gerenciador de Objetos do servidor, que é notificado das alterações realizadas e se encarrega de replicá-las aos demais componentes. Diversos componentes, inclusive de natureza diferente, compartilham os mesmos objetos de cooperação. A arquitetura provê recursos para a propagação de alterações e o controle de concorrência, livrando os desenvolvedores destas tarefas.
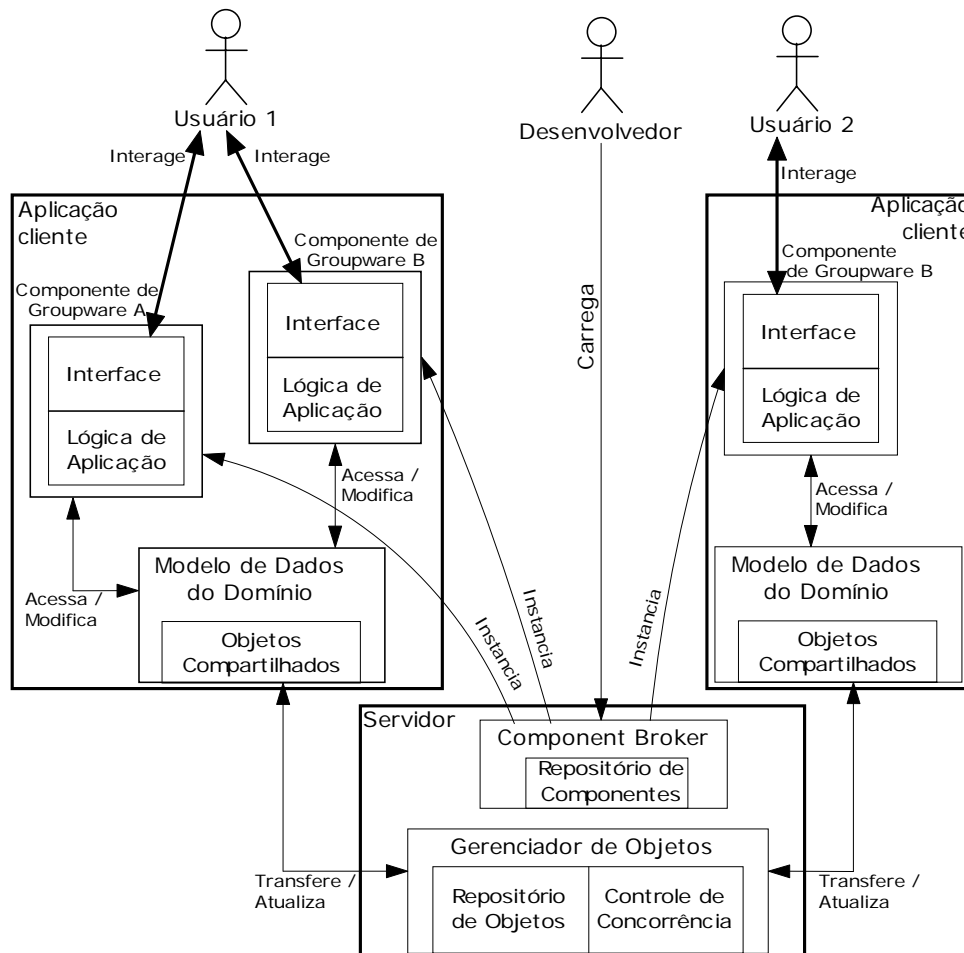
Figura 2.5. Arquitetura proposta por Tietze [2001]

A arquitetura também provê recursos para que os desenvolvedores insiram ou atualizem componentes no sistema, sem reiniciá-lo. Esta carga é feita no *Component Broker*, que é a partir de onde os clientes instanciam os componentes. O *Component Broker* encarrega-se de manter persistentes e consistentes as versões utilizadas pelos clientes, além de gerenciar as instâncias que executam no servidor. A utilização de uma arquitetura que gerencia o compartilhamento de objetos e o armazenamento e instanciação de componentes instrumentam o desenvolvedor, que por sua vez se concentra no desenvolvimento do componente como se fosse uma aplicação mono-usuário utilizando objetos locais [Tietze, 2001].

A divisão da implementação do componente em interface e lógica da aplicação possibilita que ele tenha diferentes versões da camada de interface para, por exemplo, computadores pessoais, computadores de mão e celulares. Estas implementações da camada de interface utilizam a mesma camada de lógica de aplicação, e com isto, o comportamento do componente é mantido para as

diferentes formas de acesso. Alterações no comportamento do componente se refletem nas diferentes formas de acesso.

Dewan [1998] propõe uma arquitetura genérica para sistemas colaborativos. De acordo com esta arquitetura, um groupware é estruturado em várias camadas com diferentes níveis de abstração, conforme ilustrado na Figura 2.6. A camada mais alta está relacionada ao nível semântico, específico ao domínio em questão. A camada inferior corresponde ao nível do hardware. Conforme ilustrado na figura, algumas camadas são compartilhadas e outras são replicadas nas ramificações correspondentes aos usuários conectados. As camadas comunicam-se entre si através de eventos de *feedback*, que transitam na mesma ramificação, e eventos de *feedthrough*, que são transmitidos à camada correspondente em outra ramificação.
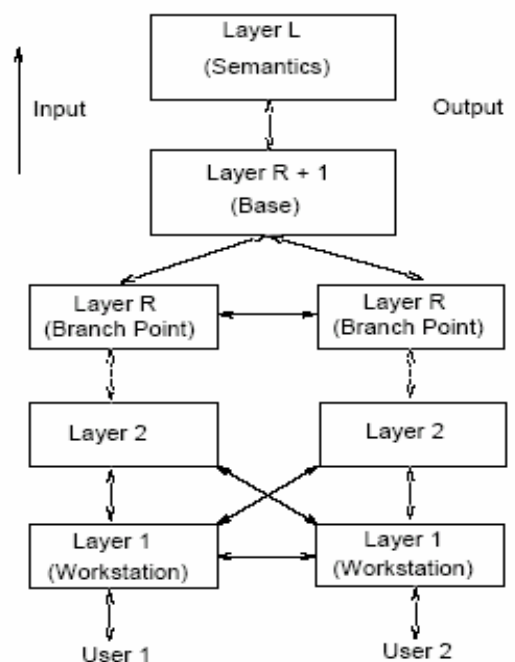
Figura 2.6. Arquitetura genérica de groupware proposta por Dewan [1998]

Laurillau & Nigay [2002] propõem uma arquitetura genérica para groupware, baseada na arquitetura de Dewan e no estilo arquitetural PAC*. O PAC* [Calvary et al., 1997] é uma extensão do PAC (*Presentation, Abstraction and Control*), que organiza a arquitetura em apresentação, que contém o código responsável pela montagem da interface com o usuário; abstração, que contém o núcleo funcional; e controle, que registra e gerencia as interdependências. No PAC*, além destas divisões, a arquitetura é separada de acordo com as classes de

funcionalidade do modelo Clover: produção, coordenação e comunicação. A Figura 2.7 apresenta o Clover Architecture.
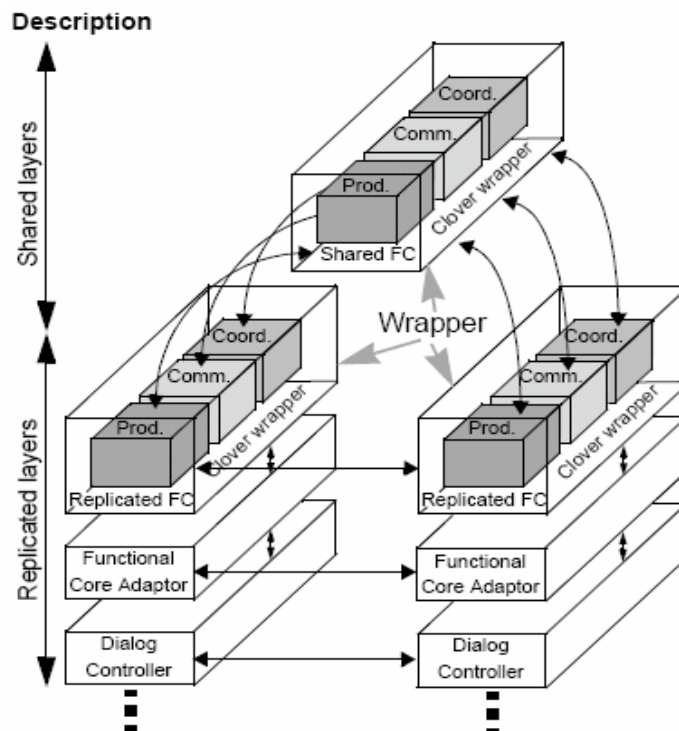
Figura 2.7. Clover Architecure [Laurillau & Nigay, 2002]

O PAC* e o Clover Architecture, assim como a abordagem proposta nesta tese, organizam a arquitetura em três classes de funcionalidades derivadas de um modelo de colaboração. Entretanto, o PAC* e o Clover Architecture oferecem um estilo arquitetural e uma arquitetura genérica para a construção de groupware, não pressupondo componentes de software compatíveis com um modelo de componentes. Nesta tese, os componentes são associados a *component frameworks* específicos. Além disto, nesta tese, tanto o groupware quanto os serviços são componentizados em função do modelo 3C, e o modelo é utilizado na organização de kits de componentes, que são construídos a partir de uma engenharia de domínio.

2.1.5. Frameworks

Um framework agiliza o processo de desenvolvimento de software que lida com determinado tipo de problema, provendo recursos que fornecem flexibilidade para os desenvolvedores adequarem-no às suas necessidades, reusando a solução

da parte comum dos problemas. De acordo com Johnson [1997], um framework é definido como um projeto reusável de todo ou de parte de um sistema, fornecendo um conjunto de classes abstratas e a forma com que suas sub-classes interagem. Um framework é um esqueleto de um sistema, que é instanciado e especializado para gerar uma família de aplicações [Govoni, 1999].

Na literatura são encontrados diversos frameworks voltados para o desenvolvimento de groupware. Prakash & Knister [1994] propõem um framework para gerência de operações em uma ferramenta colaborativa, possibilitando registrar e desfazer operações. Lee et al. [2002] propõem um framework voltado para o desenvolvimento de CVEs (*Collaborative Virtual Environments*). Kirsch-Pinheiro et al. [2002] propõem um framework para gerenciar as informações de percepção de um groupware. Nunamaker et al. [2001] propõem um framework para a gestão do conhecimento em ambientes colaborativos. Buzko et al. [2000] propõem um framework voltado para utilização da computação móvel em ambientes colaborativos. Osuna & Dimitriadis [1999] propõem um framework para o desenvolvimento de ambientes educacionais baseados na teoria do construtivismo social.

2.1.6. Avaliação Heurística

Para testar a usabilidade de um software, são utilizadas técnicas de avaliação heurística, onde um grupo de avaliadores inspeciona a interação no software identificando problemas de usabilidade [Nielsen, 1994]. Baker et al. [2001] propõem um conjunto de heurísticas específicas para sistemas colaborativos. As heurísticas são: prover meios para a comunicação verbal e intencional; prover meios para a comunicação gestual e intencional; prover meios para transmitir informação não-intencional; prover meios para transmitir informação sobre as ações nos objetos compartilhados; prover proteção de acesso; gerenciamento de diferentes níveis de acoplamento no trabalho conjunto; possibilitar a coordenação; e facilitar encontros entre os participantes.

Araujo et al. [2004] propõem um ambiente para avaliação de protótipos de groupware. A avaliação é conduzida com base em quatro dimensões principais:

contexto do grupo, colaboração obtida enquanto o grupo trabalha, usabilidade do groupware e impactos culturais provocados na organização. Uma ontologia referente aos conceitos principais da avaliação guia o processo.

As abordagens apresentadas ao longo desta seção e a componentização de groupware não são excludentes. No Capítulo 6 é discutido como agregá-las e direcioná-las a uma engenharia de groupware baseada no modelo 3C de colaboração.

2.2. Componentes de Software

Nesta seção, são apresentados conceitos do desenvolvimento baseado em componentes (DBC), resumindo o Apêndice A. No DBC, componentes de software substituíveis, reusáveis e interoperáveis são utilizados para compor a aplicação final [Gimenes & Huzita, 2005]. Um componente de software é [D'Souza & Wills, 1998, p.387]:

> Um pacote coerente de software que (a) pode ser desenvolvido e instalado independentemente como uma unidade, (b) tem interfaces explícitas e bem definidas para os serviços que provê, (c) tem interfaces explícitas e bem definidas para os serviços que espera de outros, e (d) pode ser utilizado para composição com outros componentes, sem alterações em sua implementação, podendo eventualmente ser customizado em algumas de suas propriedades.

Um componente de software é instalado em uma plataforma de execução e segue um modelo de componentes [Szyperski, 1997]. É concebido para ser autocontido, reusável e substituível e prover serviços específicos de uma maneira coesa e bem definida. Os principiais benefícios da utilização de componentes são a manutenibilidade, reuso, composição, extensibilidade, integração, escalabilidade, entre outros [D'Souza & Wills, 1998, p.397].

Para acessar e interconectar componentes, são utilizadas suas portas [OMG, 2005]. Uma porta é um meio identificável de conexão, por onde um componente oferece seus serviços ou acessa os serviços dos outros [D'Souza & Wills, 1998, pg 410]. As portas são ligadas através de conectores, implementados através de chamada de métodos, propagação de eventos, fluxo de dados, transferência de arquivos, etc. [D'Souza & Wills, 1998, p.389]. Os tipos de conectores variam para

cada tecnologia e possibilitam a conexão em tempo de codificação, compilação, inicialização ou execução.

A interface é o contrato de utilização do componente [Szyperski, 1997]. Respeitando-se os contratos, pode-se alterar a implementação interna do componente ou substituí-lo por outro, sem modificar seus clientes. A interface define as maneiras de utilizar o componente, separando a especificação da implementação. Um componente apresenta múltiplas interfaces correspondendo aos conjuntos de serviços que visam diferentes necessidades dos clientes [D'Souza & Wills, 1998, p.397]. Normalmente, o componente possui pelo menos uma interface relativa aos serviços disponibilizados (interface de negócio) e outra à conexão com a infra-estrutura de execução (interface de sistema), onde são tratados serviços técnicos, como os relacionados ao ciclo de vida, à instalação e à persistência.

As interfaces são classificadas em fornecidas (*provided interfaces*) e requeridas (*required interfaces*) [Councill & Heineman, 2001, p.9]. Um componente possui uma interface fornecida ao implementar todas as operações definidas naquela interface e uma interface requerida ao usar pelo menos uma operação definida na interface. Na UML 2.0, interfaces fornecidas são representadas por uma circunferência fechada, enquanto as interfaces fornecidas são representadas por uma semicircunferência [OMG, 2005]. Conforme ilustrado na Figura 2.8, componentes se conectam por meio da interface requerida de um com a interface fornecida de outro [Barroca et al., 2005, p.6]. Para conectar componentes com conectores incompatíveis, desenvolve-se um código adicional chamado de adaptador, que faz as conversões e operações necessárias para compatibilizar interfaces.
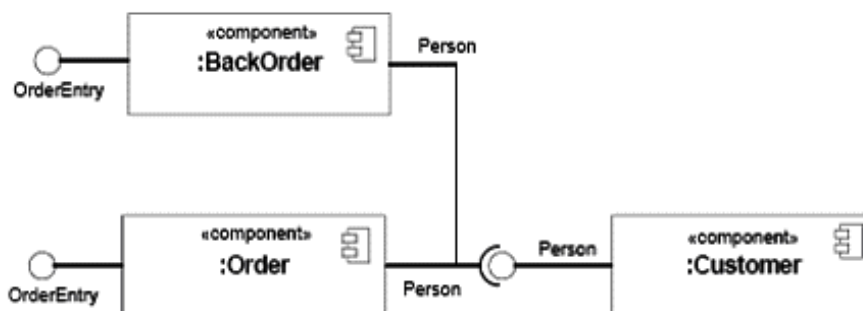
Figura 2.8. Exemplo de conexões entre componentes [OMG, 2005]

O modelo de componentes (*component model*) define vários aspectos da construção e da interação dos componentes, entre eles, a forma de implementar as interfaces e os conectores. Vários modelos apóiam-se na orientação a objetos para a implementação de interfaces e mensagens, entretanto, esta tecnologia não provê suporte à representação de interfaces requeridas e aspectos não-funcionais [D'Souza & Wills, 1998, p.388]. O modelo de componentes define também o padrão de nomeação dos componentes, de composição, de versionamento e de empacotamento. O empacotamento possibilita que um componente seja instalado como uma unidade, contendo arquivos, módulos, código executável, código fonte, código de validação, etc. [Szyperski, 1997, p.276; D'Souza & Wills, 1998, p.386].

Ao implantar o componente (*deployment*), ele é customizável [Heineman, 2000]. A customização é a habilidade de adaptar um componente antes de sua instalação ou uso, normalmente com o objetivo de especializar seu comportamento [Weinreich & Sametinger, 2001, p.42]. Na customização por composição, um componente repassa a outros as chamadas das operações. Na customização por alteração de propriedades, um arquivo descritor é utilizado para configurar o componente [Szyperski, 1997, p.33]. O arquivo descritor provê informações sobre o conteúdo do pacote, sobre as dependências externas e sobre as configurações do componente. Este arquivo é utilizado pela infra-estrutura de execução para instalar e configurar o componente [Weinreich & Sametinger, 2001, p.44].

Ao implementar um componente através da orientação a objetos, as interfaces definem o modelo de objetos compatível. Os objetos são passados de componente em componente, de modo que um componente não tem ciência da origem dos dados e do código sendo executado [Szyperski, 1997, p.42]. Chama-se de instância de componente, o conjunto de objetos pelos quais se manipula o componente [Szyperski, 1997, p.370]. Estes objetos são a manifestação do componente em tempo de execução [D'Souza & Wills, 1998, p.390].

Um *component kit* é um conjunto de componentes interoperáveis aderentes a uma padronização [D'Souza & Wills, 1998, p.404]. De um kit de componentes gera-se uma família de aplicações, fazendo diferentes arranjos e eventualmente desenvolvendo alguns sob medida [Wills, 2001, p.309]. Para desenvolver um

component kit, são analisadas aplicações similares e são identificados e generalizados componentes comuns [D'Souza & Wills, 1998, p.385].

Um *component framework* é um conjunto de interfaces e regras de interação que possibilitam a implantação de componentes aderentes a um certo padrão [Szyperski, 1997, p.26, p.280]. O *component framework* estabelece as "condições ambientais" e regula a interação entre as instâncias dos componentes.

Um container é uma plataforma, normalmente desenvolvida por terceiros, com o objetivo de hospedar e gerenciar componentes de um determinado modelo, provendo serviços de infra-estrutura de execução, como gerenciamento de transações distribuídas, pooling de recursos, acesso concorrente, segurança, persistência, etc. [D'Souza & Wills, 1998, p.401]. Os serviços providos pelo container e pelos *component frameworks* liberam os desenvolvedores de implementar serviços técnicos de baixo nível, de modo que direcionem seus esforços para as regras de negócio e para a composição do sistema. Alguns containeres possibilitam separar aspectos não relacionados à lógica da aplicação, possibilitando modificá-los sem alterar o componente. O container define uma interface que estabelece a conexão com os componentes, chamada de *lifecycle interface*, que é acessada pelo container para gerenciar a inicialização, execução e ativação do componente [Schwarz et al., 2003].

A arquitetura da aplicação descreve as propriedades, restrições e relacionamentos de suas partes [Stafford & Wolf, 2001, p.373]. O desenvolvimento baseado em componentes considera pelo menos duas visões da arquitetura: arquitetura de aplicação e arquitetura técnica [D'Souza & Wills, 1998, p.483]. Na arquitetura de aplicação são definidas a função de cada componente no contexto do sistema e a interação entre eles, de forma independente da tecnologia de suporte. A arquitetura técnica trata a tecnologia de suporte, independentemente do domínio da aplicação.

O conceito de frameworks está relacionado ao de componentes; são conceitos complementares que contribuem para o reuso de software [Gimenes & Huzita, 2005]. Os frameworks são voltados para um domínio específico de aplicação ou para a solução de problemas ligados à tecnologia. Um framework normalmente é orientado a objetos, composto de classes concretas e abstratas, interfaces e arquivos de configuração. Para especializar o framework utiliza-se

herança, composição ou configuração. Os pontos de extensão do framework são chamados de *hot-spots* [Johnson, 1997].

2.2.1. Benefícios e Dificuldades da Componentização de Software

Para projetar um sistema baseado em componentes é necessário entender os benefícios e dificuldades da tecnologia, além do ferramental disponível. Os benefícios da componentização estão ligados a manutenibilidade, reuso, composição, extensibilidade, integração, escalabilidade, entre outros [D'Souza & Wills, 1998, p.397]. As dificuldades são separadas em dificuldades do desenvolvimento para componentização (construção dos componentes e da infra-estrutura) e dificuldades do desenvolvimento com componentização. As dificuldades do desenvolvimento para componentização estão ligadas ao esforço inicial de análise, projeto e desenvolvimento, enquanto as dificuldades do desenvolvimento com componentização estão ligadas ao esforço despendido no entendimento dos componentes e das ferramentas envolvidas, à perda de flexibilidade, à dependência de terceiros e à adaptação do processo de desenvolvimento.

Uma das principais motivações para se desenvolver um software baseado em componentes é sua manutenção. Os componentes são substituíveis para atualização ou correção, muitas vezes sem precisar alterar ou recompilar a aplicação como um todo [D'Souza & Wills, 1998]. Componentes são adicionados, removidos ou substituídos por versões mais robustas ou mais apropriadas ao hardware, ao sistema operacional ou aos produtos legados com os quais o sistema tenha que operar. A modularidade obtida com a componentização facilita a localização do código a ser alterado e o encapsulamento da alteração. Algumas mudanças não acarretam em alterações no código, sendo resolvidas por re-composição da aplicação ou re-configuração dos componentes [D'Souza & Wills, 1998, p.397].

O reuso também é freqüentemente citado como um benefício da componentização. O reuso favorece a redução dos esforços de desenvolvimento e a qualidade do produto final, por colocar em uso código já utilizado e testado em

outras situações [Krueger, 1992]. Blocos com granularidade baixa (como uma classe) e alta (como um sistema) são difíceis de reusar, pois são genéricos ou específicos demais. O desenvolvimento baseado em componentes trabalha com blocos com granularidade média, mais propícia para o reuso. A componentização também promove o reuso nas diversas atividades do desenvolvimento: análise, projeto, implementação e testes.

Com a componentização, a aplicação é adaptável para diversas necessidades, selecionando e configurando os componentes mais adequados. Pode-se reimplementar um determinado componente para atender a uma necessidade específica ou adicionar novos componentes ou interfaces para estender os serviços providos, tornando o software desenvolvido mais adaptável e extensível [D'Souza & Wills, 1998, p.397].

Uma outra vantagem da componentização é o encapsulamento de conhecimento e uma programação de alto nível. Um desenvolvedor não precisa conhecer os detalhes de implementação dos componentes para utilizá-los para compor as aplicações. Quem integra componentes se especializa nesta atividade abstraindo os detalhes de implementação, tendo uma visão mais abrangente e mais próxima do domínio de aplicação.

A componentização também facilita a prototipação, pois o sistema é recomposto para experimentar idéias. Esta capacidade é especialmente útil quando há pressão para liberar produtos no mercado ou em sistemas com requisitos não definidos e instáveis [Pressman, 2000]. Com a componentização, é possível usar um ambiente RAD (*Rapid Application Development*) ou um toolkit para o desenvolvedor prototipar sua aplicação. A prototipação e o desenvolvimento iterativo possibilitam colocar o sistema em produção mais cedo, de modo a refinar gradualmente os requisitos e construir o sistema com base no aprendizado obtido com a realimentação [Teles, 2004].

A decomposição do sistema possibilita a definição de componentes independentes, que podem ser subcontratados ou alocados para outras equipes, o que favorece o desenvolvimento paralelo e em grupo. Em alguns sistemas [Won et al., 2005; Slagter & Biemans, 2000; Li & Muntz, 1998; Hummes & Merialdo, 2000], os próprios usuários finais recompõem e re-configuram os componentes, adaptando a aplicação para suas necessidades específicas. A aplicação é

incrementada para acompanhar as características das tarefas e para prototipar e experimentar configurações antes de solicitar um desenvolvimento completo.

O desenvolvedor de um componente é beneficiado pela infra-estrutura de execução, que provê serviços básicos como persistência, interconexão, escalabilidade, etc., aliviando a necessidade de implementar estes serviços. Algumas infra-estruturas possibilitam a integração de forma transparente para o programador de componentes desenvolvidos e disponibilizados em diferentes linguagens de programação, tecnologias e plataformas [D'Souza & Wills, 1998, p.397].

Com relação às dificuldades, o desenvolvimento baseado em componentes demanda um esforço inicial maior de projeto e implementação para montar a infra-estrutura do sistema e construir uma biblioteca robusta de componentes reusáveis. Projetar e preparar um pedaço de software para futuro reuso aumenta a necessidade de flexibilidade, documentação, estabilidade e abrangência do software [Moore & Bailin, 1991]. O software deve ser bem documentado, testado e deve ter um esquema robusto de validação [Pfleeger, 2001]. Os componentes não podem ser nem muito genéricos e nem muito específicos [Oliveira, 2001].

O custo do estudo e entendimento de como usar e instalar os componentes também é outra dificuldade associada com o reuso. Às vezes é mais rápido desenvolver um componente do que procurar por um pronto, estudá-lo e adaptá-lo [Pfleeger, 2001]. A menos que o custo de aprendizagem seja amortizado por vários projetos ou que o ganho de produtividade e qualidade sejam expressivos, o investimento inicial não se torna atraente [Oliveira, 2001].

O reuso de componentes provenientes de terceiros pode levar a situações inesperadas, onde se torna necessário reimplementar uma grande quantidade de código, quando novos requisitos levarem a alterações ou customizações não possibilitadas pelo componente. Os componentes introduzem dependências fora do controle dos desenvolvedores do sistema, impondo um esforço continuado de atualização de suas versões e de reconfiguração do sistema. Há um risco de incorporar bugs de terceiros no software. Além disto, muitas vezes é difícil encontrar um componente que atenda plenamente às funcionalidades desejadas e ainda ofereça suporte aos requisitos não funcionais da aplicação, como performance, segurança, escalabilidade, etc. [Gimenes & Huzita, 2005]. Por fim, o

desenvolvimento baseado em componentes também demanda uma adaptação no processo de desenvolvimento, que passa a incluir etapas como análise de domínio, busca de componentes e testes específicos.

A tecnologia de componentes vem sendo constantemente utilizada no desenvolvimento de groupware, conforme é ilustrado na próxima seção. A manutenibilidade, reuso, composição, extensibilidade, integração e escalabilidade obtidos com a componentização normalmente superam as dificuldades advindas da tecnologia e são importantes no desenvolvimento de sistemas colaborativos. As dificuldades da componentização aplicadas à abordagem e à arquitetura propostas nesta tese são retomadas no Capítulo 5.

2.3.
Groupware Baseado em Componentes

Na literatura, a tecnologia de componentes é vista como apropriada para o desenvolvimento de groupware e há diversos sistemas e plataformas que disponibilizam blocos modulares e relativamente independentes para a construção das aplicações colaborativas. Os sistemas baseados em componentes são classificados em estritamente baseado em componentes (*strictly component-based*) ou fracamente baseado em componentes (*loosely component-based*) [Slagter & Hofte, 1999]. O primeiro contempla os sistemas onde um modelo de componentes é utilizado e o segundo enfoca sistemas onde os componentes são vistos como partes independentes, sem uma padronização específica. A utilização de um modelo de componentes favorece a integração, o reuso e a substitutibilidade dos componentes, bem como a possibilidade de extensão da solução por terceiros [D’Souza & Wills, 1998]. Por sua maior relação com a proposta desta tese, na seqüência são descritos sucintamente alguns sistemas de groupware estritamente baseados em componentes.

2.3.1.
Live

Live [Banavar et al., 1998] é uma plataforma que oferece suporte a construção de groupware síncrono. O modelo de componentes do LIVE é baseado

na especificação JavaBeans e fornece uma interface de alto nível para o desenvolvimento de groupware. Os componentes do Live são baseados nos conceitos de sessões e objetos. Os objetos são compartilhados no contexto de uma sessão, que é implementada no servidor. Quatro tipos de objetos compartilhados são disponibilizados: *stateful objects*, que mantém o estado persistente; *media stream objects*, que são utilizados para prover a transmissão de áudio e vídeo no modo contínuo; *stateless objects*, que são utilizados para prover notificação de eventos e sincronização; e *token objects*, que são utilizados para prover controle de concorrência. A plataforma Live oferece serviços específicos para cada um dos tipos de objetos.

Os componentes disponibilizados na plataforma Live são predominantemente não-visuais, ou seja, não é provida uma interface com usuário para estes componentes. Entretanto, a plataforma provê funcionalidades específicas para a integração de componentes visuais provenientes de terceiros. Alguns exemplos de funcionalidades providas pelos componentes da plataforma Live são: gerenciamento de sessão, gerenciamento de convites, compartilhamento de dados e eventos, sincronização de acesso, transmissão contínua e arquivamento e recuperação de objetos e eventos.

A diferença fundamental entre os componentes da plataforma Live e os componentes propostos nesta tese está na concepção e enfoque. Os componentes desta tese são derivados de uma engenharia do domínio guiada pelo modelo 3C de colaboração, que apóia as diversas etapas do ciclo de desenvolvimento de groupware. O enfoque dos componentes da plataforma Live é em groupware síncrono, com grande parte das funcionalidades voltadas para transmissão contínua de áudio e vídeo. Os componentes desta tese, assim como os componentes da plataforma Live, foram propostos com o intuito de encapsular a complexidade de baixo nível inerente a sistemas colaborativos, com o objetivo de reduzir as dificuldades de desenvolvimento de groupware.

2.3.2. DISCIPLINE

DISCIPLINE (DIstributed *System for Collaborative Information Processing and LEarning*) [Marsic, 1999] é uma plataforma voltada para o desenvolvimento de groupware síncrono para o domínio educacional. A arquitetura do DISCIPLINE consiste de componentes replicados e recursos centralizados no servidor. Os elementos principais desta arquitetura são um barramento de colaboração, uma interface com usuário multi-modal, lógica da aplicação e agentes inteligentes.

O barramento de colaboração interconecta os elementos da arquitetura, propagando eventos e controlando a concorrência de acesso e a sincronização entre as aplicações, bem como o gerenciamento das informações de percepção. A interface com o usuário multi-modal provê diversas formas de interação entre os usuários e a aplicação. Alguns recursos oferecidos, além dos recursos de entrada e de saída textual, são o reconhecimento e a sintetização de voz e o reconhecimento de gestos. Recentemente, também foi incorporado na plataforma o suporte a construção de interfaces para dispositivos móveis, como PDAs [Krebs et al., 2003]. A lógica da aplicação é provida por componentes JavaBeans, e agentes inteligentes são utilizados para coordenar os elementos da arquitetura.

O enfoque da plataforma DISCIPLINE é no desenvolvimento de groupware síncrono com integração entre diversas ferramentas e mecanismos de entrada e de saída. Nesta tese, a abordagem proposta é direcionada para o modelo de negócio da aplicação, que é organizada em função do modelo 3C. Com relação à interface com o usuário, o suporte computacional à colaboração é harmonicamente combinado de modo a facilitar a manipulação das ferramentas e informações [Laurillau & Nigay, 2002].

2.3.3. FreEvolve

O FreEvolve [Won et al., 2005], chamado anteriormente de EVOLVE [Stiemerling et al., 1999] (antes de sua liberação na licença GPL), é um sistema baseado em componentes desenvolvido em uma arquitetura cliente-servidor na Internet. O FreEvolve foi concebido visando a adaptação e a montagem pelos usuários finais da aplicação. O modelo de componentes utilizado no FreEvolve é chamado FlexiBeans, e é uma extensão do JavaBeans. O suporte específico a portas, objetos compartilhados e interações remotas é parte das extensões do FlexiBeans. O modelo de componentes contempla tanto componentes visíveis na interface com o usuário, quanto componentes sem apresentação. No FreEvolve, a estrutura da aplicação colaborativa é descrita por arquivos que descrevem a composição da estrutura no lado servidor e no lado cliente, as interconexões entre os componentes e a ligação entre as ferramentas e os usuários. A composição pelo usuário final é alcançada a partir da manipulação destes arquivos.

O sistema FreEvolve oferece também uma API (Tailoring API) que disponibiliza recursos de customização e composição. Posteriormente, foi incorporada também no sistema uma interface gráfica 3D para manipulação e configuração dos componentes [Stiemerling et al., 2001]. O cliente do sistema FreEvolve pode ser uma aplicação stand-alone ou embutida em um navegador web. A versão embutida possui uma instalação mais direta, porém a versão stand-alone possui mais funcionalidades, visto que não está sujeita às mesmas restrições de segurança presentes na versão para navegadores.

O enfoque do FreEvolve é na geração de groupware extensível, apto a acompanhar a evolução dos processos de trabalho. A abordagem do FreEvolve é similar à arquitetura proposta nesta tese no aspecto de encapsular a complexidade da montagem da aplicação colaborativa através da utilização de arquivos para manipulação dos componentes do sistema. O uso de arquivos explicita a estrutura e a configuração do sistema e dispensa a utilização de uma ferramenta específica para a composição e customização do ambiente. Entretanto, assim como nas plataformas anteriores, o FreEvolve não utiliza explicitamente um modelo de colaboração para a concepção dos componentes e para a montagem do ambiente colaborativo.

2.3.4. DACIA

A plataforma DACIA (*Dynamic Adjustment of Component InterActions*) [Litiu & Prakash, 2000] é voltada para o desenvolvimento de groupware para dispositivos móveis. A plataforma define seu próprio modelo de componentes. Neste modelo, um componente é chamado de PROC (Processing and ROuting Component). O modelo de componentes define uma maneira particular de troca de mensagens através das portas de entrada e de saída dos componentes. Os componentes transformam os fluxos de entrada, sincronizam os fluxos de entrada e de saída e, eventualmente, direcionam partes dos fluxos de entrada para múltiplos destinos.

Na arquitetura DACIA, há um elemento chamado Engine que é executado em cada estação e é responsável pelo gerenciamento dos componentes, mantendo uma lista dos PROCs e de suas conexões. Este elemento é responsável por estabelecer e manter as conexões e as comunicações entre as estações e seus componentes. A arquitetura DACIA também prevê o uso de monitores. Um monitor trabalha em conjunto com um Engine e é responsável pelo gerenciamento dos dados e pela configuração dos componentes. Enquanto os Engines e os PROCs são de propósito geral, os monitores são específicos da aplicação em questão.

Os princípios de projeto da plataforma DACIA são a obtenção de uma arquitetura modular, usada para construção de aplicações a partir de componentes de software que implementam operações individuais; o oferecimento de mobilidade para a aplicação e para o usuário; o encapsulamento das complexidades inerentes ao desenvolvimento de aplicações para dispositivos móveis, como o gerenciamento de uma conexão intermitente; o oferecimento de suporte à conexão entre componentes remotos; e a possibilidade de reconfiguração da aplicação em tempo de execução.

Atualmente, considerar os dispositivos móveis no desenvolvimento de groupware é fundamental, dada a disseminação de PDAs e celulares com capacidade de processamento. A partir de 2004 foi dado início ao

desenvolvimento do AulaNetM, uma extensão do AulaNet para equipamentos móveis [Filippo et al., 2005]. O enfoque da arquitetura e da componentização propostas nesta tese é voltado para a camada de negócios da aplicação, prevendo a possibilidade de mais de um meio de interação. Deste modo, a lógica do suporte computacional à colaboração é reusada para uma camada de visão que funciona através da web, de aplicação stand-alone ou de dispositivos móveis. Diferentemente dos componentes previstos na arquitetura DACIA, os componentes desta tese são desenvolvidos e organizados seguindo uma abordagem 3C.

2.3.5. DreamTeam

O DreamTeam [Roth & Unger, 2000] é uma plataforma baseada em componentes para a montagem de groupware síncrono. A plataforma disponibiliza um ambiente de desenvolvimento, com ferramentas específicas, um ambiente de execução e um ambiente de simulação. São disponibilizados ao desenvolvedor componentes de groupware, componentes de interface com o usuário e componentes de manipulação de dados. Os componentes são chamados de TeamComponents e são associados aos componentes de interface com o usuário e de manipulação de dados. Os componentes TeamComponents são desenvolvidos em Java, porém seguem um modelo de componentes proprietário.

Na arquitetura do DreamTeam, os componentes não se comunicam diretamente uns com os outros. As mensagens são roteadas através de uma interface da aplicação. O modo de comunicação pode ser “intra-site”, onde um componente se comunica com outro da mesma aplicação, e “inter-site” onde um componente comunica-se com componentes de outra aplicação. O primeiro modo utiliza mensagens Java e o segundo, um mecanismo próprio de propagação de eventos. A interface dos componentes provê métodos para acessar os atributos do componente, como sua configuração, modo de integração e estado. Há também funcionalidades específicas para notificação de eventos.

Os componentes são integráveis dinamicamente à aplicação. A interface com o usuário do componente se torna parte da interface da aplicação em uma

janela existente ou como uma nova janela. O componente é configurável para diferentes modos de colaboração: privado, exclusivo ou compartilhado. Um componente sendo executado no modo privado não possui dados compartilhados; no modo exclusivo, os dados são modificáveis por um usuário de cada vez; e no modo compartilhado, vários usuários manipulam simultaneamente os dados. O DreamTeam possui diversos mecanismos para controle de concorrência, como o gerenciamento de transações envolvendo a aplicação, os recursos e a definição de políticas de acesso para os componentes e seus métodos.

O enfoque do modelo de componentes do DreamTeam é o suporte à conexão entre componentes distribuídos, de uma maneira particular para groupware. Na arquitetura proposta nesta tese não há a preocupação com a distribuição dos componentes, pois a maior parte deles é executada no servidor. Quando a comunicação remota é necessária, são utilizados os mecanismos padrões da plataforma J2EE (http://java.sun.com/j2ee).

2.3.6. IRIS

IRIS [Koch & Koch, 2000] é uma aplicação para edição colaborativa de documentos multimídia, construída a partir de componentes que utilizam um mecanismo de persistência distribuído e serviços voltados ao gerenciamento de informações de percepção. O IRIS oferece funcionalidades para editar e exibir a estrutura e o conteúdo de documentos, e para contextualizar o participante através de informações, como quem está trabalhando com o documento, quem está disponível, etc.

Os componentes da aplicação são chamados de *tools*. Para oferecer um ambiente de execução aos componentes, é disponibilizado na plataforma IRIS um *component framework* chamado ICI (IRIS Component Infrastructure). Este *component framework* possibilita a integração de diferentes ferramentas e oferece serviços de execução, que facilitam o desenvolvimento de novos componentes para a plataforma. O ICI possui um gerente de componentes chamado ToolConnector que gerencia as ferramentas e oferece a elas serviços de execução.

Na arquitetura do IRIS não há componentes centrais. As ferramentas, os dados e o controle são replicados. Eventualmente, um usuário trabalha desconectado no documento compartilhado e, ao sincronizar os dados, a plataforma IRIS oferece mecanismos para identificar e tratar os eventuais conflitos e gerenciar as versões intermediárias. A plataforma também oferece recursos para importação e exportação dos dados.

O foco do IRIS é a investigação das questões de consistência e integridade em um ambiente multi-usuário distribuído, onde os participantes trabalham conectados e desconectados, sincronamente e assincronamente. O IRIS foi desenvolvido também com o intuito de investigar as informações de percepção necessárias para o trabalho em grupo. Diferentemente do IRIS, nesta tese não há uma preocupação explícita com a autoria compartilhada de documentos e o suporte computacional à percepção é distribuído no suporte computacional à comunicação, coordenação e cooperação.

2.3.7. JViews

JViews [Grundy et al., 1997] é um *component framework* voltado para a construção de sistemas colaborativos com múltiplas visões. É disponibilizado um repositório central, que é encarregado de propagar as alterações para as visões afetadas, seguindo o padrão modelo, visão e controle. As inter-relações entre os componentes são utilizadas para troca de dados, agregação de funcionalidades e mantenção da consistência geral do sistema.

O JViews também disponibiliza um modelo de eventos que é utilizado para descrever as mudanças nos componentes ou ocorrências que são tratadas externamente. Para manter a consistência entre os diversos componentes e visões, o JViews disponibiliza políticas de restrições de integridade, mecanismos de persistência, meios de propagação de eventos, componentes adaptadores, controle de versões e registro dos estados e eventos, de modo a oferecer suporte às operações de desfazer e refazer ações.

Nesta tese, as funcionalidades presentes no JViews são tratadas pelos frameworks de infra-estrutura, responsáveis pela persistência, gerenciamento dos

dados, propagação de eventos, transações distribuídas, etc. Mais detalhes da arquitetura técnica utilizada nesta tese são descritos na dissertação de Barreto [2006].

2.3.8. CoCoWare

A plataforma CoCoWare [Slagter & Biemans, 2000] oferece aos usuários finais a capacidade de compor a aplicação de acordo com suas necessidades e estendê-la para acompanhar a evolução dos processos de trabalho. O CoCoWare oferece componentes para lidar com as sessões de trabalho e para gerenciar as ferramentas colaborativas, informando o que pode ser modificado, como pode ser feito e qual o impacto das modificações. O CoCoWare é uma implementação da arquitetura genérica CooPS (Cooperative People & Systems), que define os tipos básicos de componentes para a construção de aplicações colaborativas: *conference manager*, *conference tools, coordinator* e *conference enablers* [Slagter et al., 2001].

A plataforma foi inicialmente baseada no modelo de componentes do CORBA e, atualmente, estende o modelo de componentes da arquitetura .NET da Microsoft. O CoCoWare oferece um *component framework* onde os componentes são plugados e é disponibilizado um Software Development Kit (SDK) para terceiros desenvolverem novos componentes para a plataforma ou adaptarem componentes de outros modelos (COM e ActiveX).

Diferentemente do enfoque principal do CoCoWare, a abordagem proposta nesta tese visa instrumentar o desenvolvedor de groupware. Entretanto, alguma capacidade de extensão é oferecida ao administrador do ambiente, que pode instalar novos componentes e configurá-los.

2.3.9. Habanero

O Habanero [Chabert et al., 1998] provê aos desenvolvedores ferramentas para criar aplicações colaborativas em Java. É oferecido um ferramental para criar ou migrar aplicações e applets existentes para o ambiente de groupware. O

Habanero disponibiliza um servidor que hospeda e gerencia as sessões e as conexões dos clientes. O *component framework* da plataforma disponibiliza serviços de registro de sessões, gerenciamento de participantes e controle de acesso. As aplicações clientes do Habanero são chamadas de Hablets. Através da interface provida pelo *component framework*, um Hablet cria, conecta e visualiza informações das sessões e dos participantes, gerencia e propaga eventos, ativa ferramentas e manipula o catálogo de endereços compartilhado. Alguns exemplos de ferramentas desenvolvidas para a plataforma Habanero são um navegador web compartilhado, um whiteboard, um bate papo em áudio, um visualizador cooperativo de documentos VRML e um compartilhador de área de trabalho, baseado no VNC (*Virtual Networking Computing*).

As ferramentas são replicadas nos clientes e as mudanças de estados são distribuídas. Quando um novo cliente entra em uma sessão, são enviadas informações de quais ferramentas estão ativas naquela sessão e os dados necessários para compor o estado atual da colaboração. O Habanero possibilita múltiplas sessões, sendo que um mesmo participante pode atuar em mais de uma sessão simultaneamente. Uma nova ferramenta é integrada dinamicamente a uma sessão existente. A plataforma possibilita também salvar o estado de uma sessão de modo a prosseguir nela posteriormente. No Habanero não há o conceito explícito de um modelo de colaboração. Nesta plataforma, os elementos do suporte computacional à colaboração são associados ao gerenciamento de sessões.

2.3.10. COCA

A plataforma COCA (*Collaborative Objects Coordination Architecture*) [Li & Muntz, 1998] oferece uma maneira de separar as políticas de coordenação, que normalmente ficam embutidas no código da aplicação. As políticas de coordenação são descritas em uma linguagem de definição, e o coordenador do grupo pode alterar dinamicamente estas políticas, refinando o ambiente para acompanhar a evolução da realização das tarefas. As políticas são definidas em função dos papéis dos usuários. A máquina virtual que interpreta as políticas de coordenação, chamada de *Cocavm*, é replicada nas estações dos clientes.

A arquitetura do COCA prevê uma organização em camadas, com uma camada de comunicação, de coordenação e de colaboração. A camada de comunicação oferece serviços de transmissão de dados para todas estações ou para um determinado usuário em particular. Acima desta camada, vem a camada de coordenação que interpreta as políticas de coordenação. A camada de colaboração é responsável por oferecer uma infra-estrutura de execução para as ferramentas colaborativas. A arquitetura também provê um barramento de colaboração que abstrai as funcionalidades do canal de comunicação.

Apesar da arquitetura do COCA utilizar comunicação, coordenação e colaboração, o entendimento destes conceitos difere do modelo 3C utilizado nesta tese. A camada de comunicação do COCA é voltada para a comunicação remota entre componentes em vez de participantes. A arquitetura do COCA não oferece suporte específico ao conceito de cooperação do modelo 3C. A camada de colaboração funciona como um *component framework* para os componentes de colaboração. Além disto, os conceitos adotados na arquitetura do COCA não são utilizados para organizar ou conceber os componentes de colaboração.

2.3.11. GroupKit

O GroupKit [Roseman & Greenberg, 1996] é um toolkit contendo componentes e uma plataforma de execução. O GroupKit é construído em Tcl/Tk e é voltado ao desenvolvimento de groupware síncrono. O toolkit encapsula diversas complexidades inerentes ao desenvolvimento deste tipo de aplicação, de modo que o desenvolvedor direcione seus esforços para o projeto da interação. Alguns exemplos de aplicações desenvolvidas com o kit, que são disponibilizadas juntamente com a plataforma, são: Brainstorming, que possibilita a usuários trocar idéias através de textos breves em uma área visível a todos; File Viewer, que possibilita que vários usuários visualizem um texto simultaneamente; Hello World, que disponibiliza um botão que quando pressionado, transmite uma mensagem para todos usuários conectados; Text Chat, um programa de chat semelhante ao Talk do ambiente Unix, que exibe imediatamente o que cada pessoa escreve; Tic Tac Toe, o Jogo da Velha; e Tetrominoes, para rodar e mover

diversos tipos de polígonos. A Figura 2.9 apresenta algumas aplicações desenvolvidas utilizando o GroupKit.
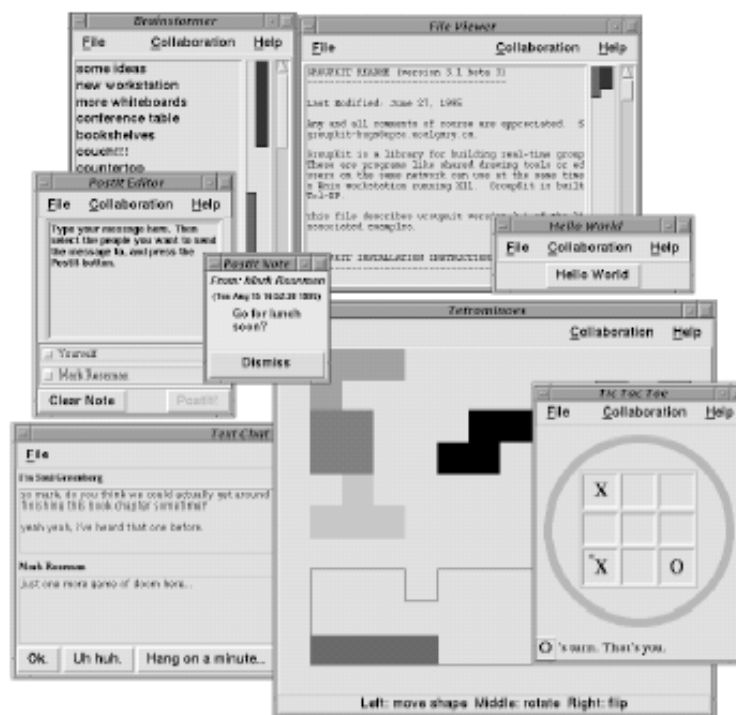
Figura 2.9. Aplicações desenvolvidas utilizando o GroupKit

O ambiente de execução do GroupKit gerencia a criação, a interconexão e a comunicação dos processos distribuídos que compõe a sessão colaborativa. O GroupKit oferece facilidades aos programadores para interconexão, gerenciamento de eventos e compartilhamento de dados. Cada estação cliente possui uma réplica do gerenciamento de sessões e da ferramenta colaborativa, enquanto o repositório fica localizado em um servidor central. O gerenciador de sessões de cada participante transmite informações sobre a sessão e sobre os participantes para o repositório central. Após o estabelecimento da conexão inicial, as ferramentas trocam informações diretamente, através de chamadas remotas de procedimento, de propagação de eventos ou de variáveis de ambiente compartilhadas entre as diversas aplicações. São oferecidos aos programadores mecanismos que encapsulam e abstraem os detalhes técnicos de conexão e comunicação.

O GroupKit oferece diversos widgets de interface que são utilizados para compor a interface gráfica da aplicação. O modelo de componentes adotado possibilita que o desenvolvedor crie novos ou estenda os widgets existentes. Há

widgets para prover informações de percepção, telepointers, barras de rolagem multi-usuário, visão de radar da área compartilhada, etc. Seus widgets são particularmente relevantes para a construção de interfaces WYSIWS (*What You See Is What I See*) relaxadas, onde as dimensões das janelas utilizadas podem ser distintas. O GroupKit foi utilizado como base para a construção de diversos sistemas colaborativos, como o Alliance [Michailidis & Rada, 1996] e o GroupCRC [Churcher & Cerecke, 1996]. Recentemente, o GroupLab[1], que desenvolve o GroupKit, disponibilizou também widgets voltados a encapsular os detalhes técnicos de interação com dispositivos físicos (phidgets), de modo que o desenvolvedor monta uma solução integrada de hardware e software [Greenberg & Fitchett, 2001]. Também foi disponibilizado um toolkit específico para SDG (*Single Display Groupware*), que objetiva oferecer suporte computacional à colaboração de vários participantes utilizando uma mesma máquina, com vários dispositivos de entrada e saída [Tse & Greenberg, 2004].

O GroupKit possui um enfoque maior na construção da interface com o usuário para groupware síncrono. Nesta tese, o enfoque é em instrumentar o desenvolvedor oferecendo componentes organizados em função do modelo 3C para construção de sistemas colaborativos na web. Apesar de o estudo de caso feito com o ambiente AulaNet possuir widgets de interface, sua construção foge do escopo desta tese.

2.3.12. Portalware

Portalware, também conhecido como CMS (*Content Management System*), é um tipo de sistema utilizado para construir portais na web. Estes sistemas apresentam uma arquitetura modular baseada em componentes para a construção de portais que oferecem suporte à interação com os usuários. Alguns exemplos de portalware são o Lumis[2], o Mambo[3], o XOOPS[4], o Zope[5] e o DotNetNuke[6]. Estes

---

[1] http://grouplab.cpsc.ucalgary.ca
[2] http://www.lumis.com.br
[3] http://www.mamboserver.com
[4] http://www.xoops.org
[5] http://www.zope.org
[6] http://www.dotnetnuke.com

sistemas oferecem flexibilidade para o administrador do portal compor as páginas incluindo ferramentas, configurando-as e posicionando-as.

Figura 2.10. Seleção de serviços no Lumis

A Figura 2.10 apresenta a tela de seleção de serviços do Lumis. Alguns dos serviços disponíveis neste ambiente são chat, e-mail, fórum de discussão, boletim, agenda de grupo e pessoal, relatórios e estatísticas, tarefas, gerenciamento de usuários e grupos, enquete, alertas, artigos, documentos, matérias e comentários. Cada serviço responde às requisições com dados no formato XML, que são convertidos para o formato HTML através de scripts XSL. Esta conversão possibilita a utilização do mesmo serviço em diversos portais ou em diferentes seções do mesmo portal, com diferentes padronizações visuais, a utilização do mesmo serviço em várias plataformas, e a criação de *skins*, que possibilitam alterar a interface do ambiente como um todo.
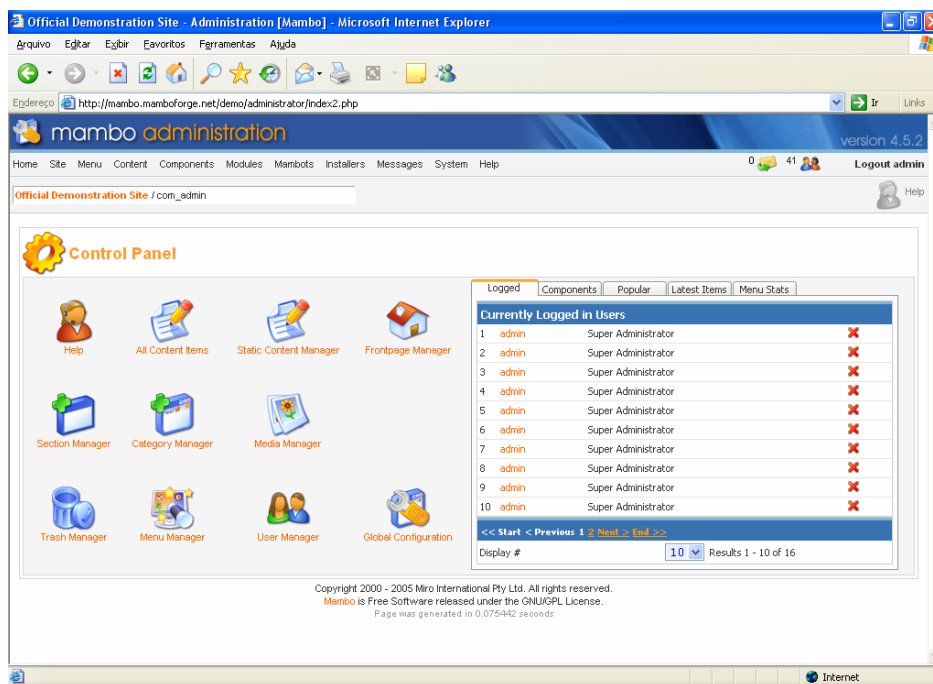
Figura 2.11. Interface administrativa do Mambo

O Mambo, cuja interface administrativa é apresentada na Figura 2.11, é desenvolvido na linguagem PHP e possui código fonte aberto. As ferramentas de colaboração do Mambo são tratadas como componentes e há um modelo de componentes que padroniza a construção de novos módulos ao ambiente. Da mesma maneira que o Lumis, o Mambo possibilita ao administrador montar as páginas, adicionando e configurando os serviços, e oferece a possibilidade de uso de *skins*. O Mambo oferece diversas opções de configuração para as páginas do portal, como tamanho, posicionamento de *banners* e estilos. Grande parte dos serviços disponíveis na plataforma é voltada para o gerenciamento de conteúdos, como documentos, matérias e notícias.
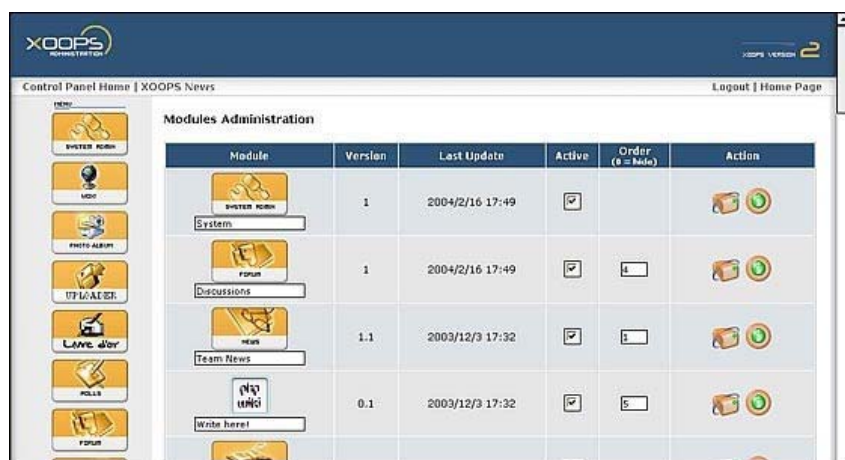
Figura 2.12. Gerenciamento de serviços no XOOPS

O XOOPS (*eXtensible Object Oriented Portal System*) também foi desenvolvido na linguagem PHP e está disponível sob a licença GNU General Public License (GPL). O site é customizado pelo administrador, que adiciona ou remove módulos através do ambiente administrativo. A Figura 2.12 apresenta a tela de gerenciamento de serviços da plataforma. Cada módulo do XOOPS oferece um conjunto de blocos pré-definidos para exibição das saídas, que são utilizados para compor a interface com o usuário. Alguns módulos já disponíveis na instalação do XOOPS são Notícias, Fórum, Enquete, Links, Downloads, Headlines, FAQ, Parceiros (banners), Usuários e Fale Conosco.

O DotNetNuke é desenvolvido para a plataforma .NET da Microsoft. Alguns serviços disponíveis na plataforma são Avisos, Banners, Newsfeeds, Fórum, FAQ, Lista de discussão, Calendário, Links, Busca e Enquetes. Assim como os demais portalware, o DotNetNuke possibilita o uso do *skins* para troca da aparência do ambiente como um todo.

A Sun Microsystems, através do Java Community Process (JCP), disponibilizou uma especificação para padronizar componentes e containeres para portais. De acordo com o modelo, os *portlets*, como são chamados os componentes, geram fragmentos de páginas que são compostas pelo container para gerar a página final. O container intermedeia a interação entre clientes e *portlets* e gerencia seu ciclo de vida. Mais detalhes sobre o modelo de componentes do *portlets* são encontrados no Apêndice A.

Nenhuma das tecnologias para construção de portais analisadas apresenta um modelo de componentes que incorpora aspectos específicos do suporte computacional à colaboração. Os modelos de componentes destas plataformas oferecem recursos para encapsular serviços técnicos e para cuidar do ciclo de vida dos componentes. Até o momento não há um padrão único para a construção de serviços para portais. Pretende-se futuramente desenvolver adaptadores do modelo de componentes utilizado nesta tese para as infra-estruturas de execução dos portalware, de modo a compatibilizar os serviços desenvolvidos para o ambiente AulaNet com os portalware e vice-versa.

2.3.13. Outras Plataformas para a Construção de Groupware

A plataforma ACOST [Hummes & Merialdo, 2000] é voltada para a criação de sistemas colaborativos extensíveis pelos usuários em tempo de execução. O modelo de componentes JavaBeans é utilizado como base para o modelo de componentes definido na plataforma. A plataforma estende o modelo de eventos definido no JavaBeans para possibilitar a troca de eventos entre componentes, mesmo eles estando distribuídos na rede. O ACOST é primordialmente utilizado para a construção de ferramentas de comunicação síncrona. Alguns exemplos de ferramentas construídas utilizando a plataforma incluem um chat e um sistema de votação online.

O Flexible JAMM (*Java Applets Made Multiuser*) [Begole et al., 1999] provê componentes alternativos para a biblioteca gráfica do Java para transformar um applet mono-usuário em uma aplicação multi-usuário, sem alterar a aplicação original. Os componentes possibilitam que diversos usuários visualizem e editem as informações, além de oferecer telepointer, visão de radar, controle de permissões e indicação de atividade. As versões mono-usuário dos componentes gráficos são substituídas dinamicamente por versões colaborativas. A substituição dos componentes é feita por uma customização na máquina virtual Java, o que reduz a sua portabilidade.

TOP [Guerrero & Fuller, 1999] é um framework em Java voltado para o desenvolvimento de aplicações colaborativas na web. Ele provê abstrações pré-definidas, que possibilitam o gerenciamento dos usuários e seus papéis, a manutenção da memória do grupo e os relacionamentos entre os participantes das sessões colaborativas. Um applet intermedeia a comunicação do cliente com o servidor, chamado de TopServer. O applet é acessível através de comandos JavaScript ou de outras aplicações Java.

Sacramento et al. [2004] propõem o middleware MoCA (*Mobile Collaboration Architecture*) voltado para oferecer suporte ao desenvolvimento de aplicações colaborativas que utilizam dispositivos móveis e informações de contexto. O MoCA oferece APIs para programação de aplicações para o lado cliente e para o lado servidor, serviços para monitoramento e inferência da

localização e contexto dos dispositivos e um framework orientado a objetos para instanciar *proxies* customizados. O middleware encapsula funcionalidades para lidar com a mobilidade do dispositivo, com a limitação de recursos e com a intermitência da conexão.

Siqueira et al. [2003] propõem uma arquitetura baseada em frameworks e componentes que possibilita instanciar ambientes educacionais para diferentes metodologias e teorias de aprendizagem, de forma a moldar um ambiente específico para cada caso. Os componentes principais da arquitetura são: *Data* e *Metadata Management*, *Groupware Management*, *Content Development Management*, *Assessment and Evaluation Management*, *Interface Management*, *Role and Security Management* e *Rule Management*.

O sistema Sieve [Isenhour et al., 1997] provê suporte à montagem de aplicações para visualização colaborativa de dados. Os componentes são interligados de modo a conectar fontes de dados e componentes de processamento e de visualização. O modelo de componentes do Sieve é uma extensão do JavaBeans. A aplicação é extensível dinamicamente através da inserção de novos componentes. Os dados são processados em um servidor central, de modo que todos os participantes visualizam o mesmo fluxo, em um estilo WYSIWIS (*What You See Is What I See*). Entretanto, os participantes podem atuar em diferentes partes das informações.

Dourish [1998] propõe a plataforma Prospero, que visa oferecer mais flexibilidade que os toolkits tradicionais, ao oferecer uma meta-arquitetura utilizada pelos desenvolvedores para definir as maneiras pelas quais os componentes são combinados, customizados e utilizados. O Prospero é implementado em CLOS (*Common Lisp Object System*) e é direcionado para encapsular as complexidades técnicas de baixo nível. O Prospero não oferece suporte à construção da interface com o usuário, pressupondo a utilização de outro toolkit com este propósito.

O COPSE-Web [David & Borges, 2004] é uma extensão do ambiente COPSE (*Collaborative Project Support Environment*) [Dias & Borges, 1999], voltado para a construção de groupware para web. O COPSE-Web adota uma arquitetura baseada no estilo MVC (Modelo-Visão-Controle). A infra-estrutura da plataforma oferece vários serviços de execução às ferramentas colaborativas e um

framework de classes a partir do qual as ferramentas são instanciadas. A infra-estrutura provê às ferramentas recursos de gerenciamento de projetos, processos, documentos, percepção e perfil. Algumas das ferramentas já disponíveis no ambiente são: quadro de avisos, relatório de eventos, fórum de discussão e agenda, oferecendo suporte a pré-reunião, reunião e pós-reunião.

Anderson et al. [2002] propõem uma abordagem para construção de toolkits denominada Dragonfly, que mantém um link bidirecional entre a arquitetura conceitual e a implementação, possibilitando ao desenvolvedor uma transição suave entre os níveis. A abordagem foi utilizada na construção do toolkit TeleComputing Developer (TCD).

Guicking et al. [2005] propõem o framework Agilo, voltado para integração de aplicações colaborativas. O framework flexibiliza a infra-estrutura de comunicação e os modelos de distribuição, de compartilhamento, de concorrência e de sincronização, oferecendo para eles implementações recorrentes.

2.4. Engenharia do Domínio e Componentes

A engenharia de domínio objetiva disponibilizar componentes que implementam os conceitos de um domínio de software em particular e possam ser reusados para implementar novas aplicações deste domínio [Werner & Braga, 2005]. Ao mapear conceitos do domínio, aumenta-se a chance de reuso em todas as fases do desenvolvimento, desde a análise até a implantação, e não é considerada somente uma única aplicação, mas sim uma família. Esta abordagem torna o grau de reuso de um dado componente maior, assim como seu entendimento, uma vez que será diretamente mapeado no domínio da aplicação (menor distância semântica) [D'Souza & Wills, 1998, p.720]. O componente é codificado (espaço da solução) de acordo com as necessidades do domínio (espaço do problema). A engenharia do domínio possibilita a uniformidade dos conceitos tratados pelos envolvidos no projeto e representados nas diversas etapas de desenvolvimento e produtos advindos do processo.

A engenharia do domínio engloba a análise do domínio, o projeto de uma arquitetura orientada ao domínio e a implementação dos componentes

correspondentes [Werner & Braga, 2005]. A análise do domínio é o processo de identificar e organizar o conhecimento sobre uma classe de problemas para apoiar sua descrição e solução [Pietro-Diaz & Arango, 1991]. Na analise do domínio é identificado, colecionado, organizado e representado o conhecimento advindo dos sistemas, da teoria de apoio, da tecnologia e do desenvolvimento de um domínio de interesse [Peterson, 1991].

A análise do domínio serve de guia para o processo de construção e reuso de componentes [Pietro-Diaz & Arango, 1991], sendo que ao reusar um elemento em um nível, o elemento correspondente no nível mais abstrato também é reusado [Werner & Braga, 2005]. Na análise do domínio, nenhuma solução de software é assumida. O propósito da modelagem é entender os conceitos e seus relacionamentos, sendo o principal produto desta atividade a definição de um modelo do domínio, que estrutura os conceitos, as regras, o vocabulário, o contexto, as funcionalidades e os relacionamentos presentes no domínio. A análise do domínio guia a forma de pensar, no que diz respeito à predição, explicação ou derivação de fatos sobre o domínio, e fornece uma classificação e organização das características dos sistemas daquele domínio [Arango, 1994].

Uma vez modelado o domínio, são desenvolvidas diversas aplicações distintas com base no mesmo modelo. A modelagem embasa a criação de ferramentas, técnicas e componentes para oferecer suporte às diversas atividades do desenvolvimento de software, instrumentando todo o desenvolvimento. A análise do domínio é feita consultando especialistas no domínio e a literatura da área ou com base no conhecimento adquirido pelos desenvolvedores durante o processo de desenvolvimento ou de utilização de diversas aplicações de um mesmo domínio. Outra fonte bastante utilizada para a análise do domínio é o estudo de aplicações desenvolvidas, no intuito de levantar as funcionalidades recorrentes, invariantes e opcionais [Werner & Braga, 2005]. A análise do domínio sistematicamente extrai características dos sistemas de uma mesma família [Griss, 2001, p.410]. O modelo fornece um vocabulário compartilhado entre os desenvolvedores e usuários, o que facilita a comunicação e o reuso, e oferece uma noção da abrangência do domínio e de suas relações [Werner & Braga, 2005].

Há vários processos de engenharia de domínio na literatura, dentre eles, o FODA (*Feature Oriented Domain Analysis*) [Kang et al., 1990], o FORM (*Feature Oriented Reuse Method*) [Kang et al., 1998] e o RSEB (*Reuse-Driven Software Engineering Business*) [Jacobson et al., 1997]. No contexto desta tese, é utilizado o processo CBD-Arch-DE [Blois et al., 2004], que é uma evolução do processo Odyssey-ED [Braga, 2000], por ser mais direcionado e instrumentado para o desenvolvimento baseado em componentes. O processo CDB-Arch-DE organiza a engenharia do domínio em 4 atividades principais: planejamento do domínio, análise do domínio, projeto do domínio e implementação do domínio, conforme ilustrado na Figura 2.13.
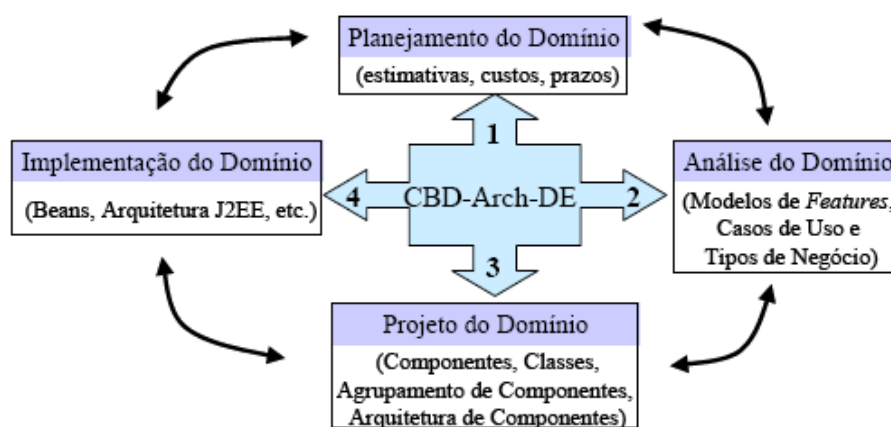
Figura 2.13. Atividades do processo CBD-Arch-DE [Blois et al., 2004]

A Figura 2.14 ilustra as atividades da análise do domínio no CBD-Arch-DE. A análise do domínio se inicia pela identificação dos contextos que representam os sub-domínios do domínio principal. Após esta atividade, são identificadas as *features* (características) correspondentes, criando um modelo de características. As características são obtidas de especialistas, usuários, documentações e de aplicações já existentes e modelam os aspectos variáveis e invariáveis do domínio, em um alto nível de abstração. As características são classificadas em conceituais, funcionais e tecnológicas e definidas como obrigatórias ou opcionais. Estas propriedades são mapeadas para os demais artefatos do domínio, que são gerados com base no modelo de características. Após a identificação das características, são identificados os tipos de negócio, que representam os aspectos estáticos do domínio candidatos a persistência no contexto do projeto, e os casos de uso associados.
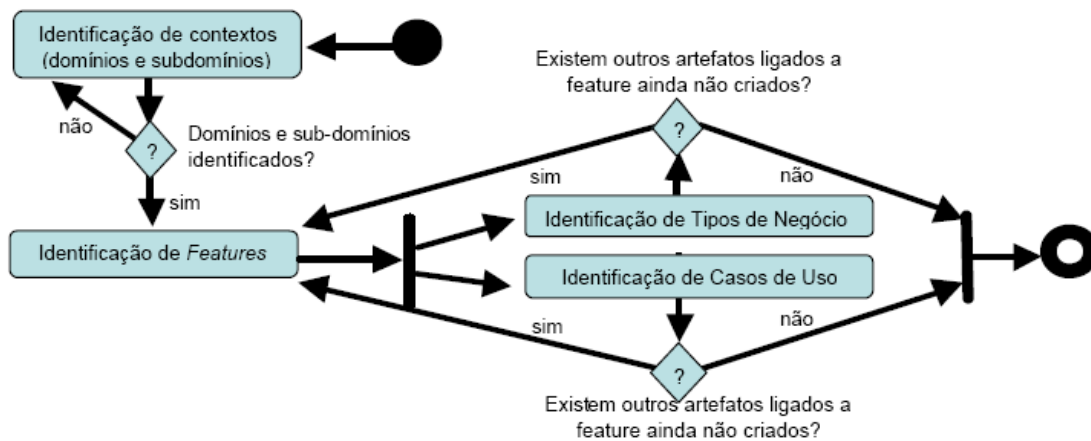
Figura 2.14. Análise do domínio no processo CBD-Arch-DE [Blois et al., 2004]

O projeto do domínio envolve a criação, composição e geração de uma arquitetura de componentes e de suas interfaces. A criação de componentes do domínio é efetuada com base em estilos arquiteturais baseado em tipos de negócio, sendo especificados o conjunto de classes que os implementam, seus métodos, atributos e relacionamentos. Após a definição dos componentes e interfaces, procura-se agrupar os componentes relacionados, por critérios de acoplamento e coesão, para obter um grau de granularidade adequado. A última atividade do processo é a implementação do domínio. Nesta etapa, é feito o mapeamento dos componentes do domínio para uma tecnologia de componentes específica.

O uso de um processo de engenharia do domínio guia a especificação dos componentes e os torna mais propícios para o reuso, por prever o domínio e os componentes nas diversas etapas do desenvolvimento do sistema. Nesta tese, o processo de engenharia do domínio é utilizado para definir o conjunto de componentes do *component kit* utilizado.

2.5. Considerações Finais

Na literatura são encontradas diversas abordagens para instrumentar o desenvolvimento de groupware. Neste capítulo foram apresentados exemplos de requisitos de groupware [Tietze, 2001; Schmidt & Rodden, 1996; Mandviwalla & Olfman, 1994], UML estendida [Rubart & Dawabi, 2002], padrões específicos [Groupware Patterns Swiki, 2005; Lukosch & Schümmer, 2004; Santoro et al.,

2001], arquiteturas de groupware [Tietze, 2001; Dewan, 1998], frameworks [Prakash & Knister, 1994; Lee et al., 2002; Kirsch-Pinheiro et al., 2002; Nunamaker et al., 2001; Buzko et al., 2000], e técnicas de avaliação de groupware [Baker et al., 2001; Araujo et al., 2004]. A engenharia de domínio, o desenvolvimento baseado em componentes e o modelo 3C de colaboração são utilizados para conectar este ferramental e torná-lo interoperável.

O desenvolvimento baseado em componentes tem se mostrado uma abordagem de desenvolvimento bastante promissora para o desenvolvimento de groupware [Blois & Becker, 2002]. Na literatura, há diversas propostas de utilização de componentes de software na construção de groupware [Banavar et al., 1998; Marsic, 1999; Won et al., 2005; Litiu & Prakash, 2000; Roth & Unger, 2000; Koch & Koch, 2000; Grundy et al., 1997; Slagter & Biemans, 2000; Chabert et al., 1998; Li & Muntz, 1998; Roseman & Greenberg, 1996; Hummes & Merialdo, 2000; Begole et al., 1999; Guerrero & Fuller, 1999; Siqueira et al., 2003; Isenhour et al., 1997]. Entretanto, nenhuma delas utiliza o modelo 3C de colaboração e uma engenharia de domínio como base para a concepção e organização dos componentes e do processo de desenvolvimento.

Szyperski [2003] afirma que há quatro motivações principais para utilizar componentes de software. A primeira e mais antiga é relacionada com a idéia de *mercado de componentes*. Nesta visão, as empresas buscam e adquirem componentes necessários para resolver problemas específicos e integram estes componentes ao sistema sendo desenvolvido. A segunda motivação é relacionada com *linha de produto*. São desenvolvidos componentes com o objetivo de reusá-los em diversos sistemas, reduzindo o custo total de investimento e os custos de manutenção. A terceira está relacionada com a idéia de composição pelo usuário final (*tailorability*). Disponibiliza-se uma infra-estrutura onde o usuário implanta componentes (*deployment*) visando estender a capacidade do sistema. A quarta motivação está relacionada com a utilização de *serviços dinâmicos*, descobertos e instalados na medida da necessidade. Nesta abordagem, ao necessitar de uma determinada funcionalidade, o sistema consulta catálogos de serviços e instala e se re-configura. A utilização desta abordagem possibilita a construção de sistemas que evoluem para acompanhar novas demandas. Web services é uma tecnologia que vem sendo utilizada com este propósito [Hansen et al., 2005].

Plataforma	Objetivo	Motivação [Szyperski, 2003]	Modelo de Componentes	Linguagem
Live [Banavar et al., 1998]	Groupware síncrono	linha de produto	Extensão do JavaBeans	Java
DISCIPLINE [Marsic, 1999]	Groupware síncrono para educação	linha de produto	Extensão do JavaBeans	Java
FreEvolve [Won et al., 2005]	Groupware distribuído	tailorability	Extensão do JavaBeans (FlexiBeans)	Java
DACIA [Litiu & Prakash, 2000]	Groupware para dispositivos móveis	linha de produto	Proprietário	Java
DreamTeam [Roth & Unger, 2000]	Groupware síncrono	linha de produto	Proprietário	Java
IRIS [Koch & Koch, 2000]	Edição colaborativa de documentos multimídia	linha de produto	Proprietário	Java
JViews [Grundy et al., 1997]	Sistemas colaborativos com múltiplas visões	linha de produto	Proprietário	Java
CoCoWare [Slagter & Biemans, 2000]	Groupware em geral	tailorability	Extensão do .Net	.Net
Habanero [Chabert et al., 1998]	Groupware síncrono	linha de produto	Proprietário	Java
COCA [Li & Muntz, 1998]	Flexibilidade na coordenação	tailorability	Proprietário	Java
GroupKit [Roseman & Greenberg, 1996]	Groupware síncrono	linha de produto	Proprietário	Tcl/Tk
Lumis (www.lumis.com.br)	Portais web	tailorability	Proprietário	Asp.NET
Mambo (www.mamboserver.com)	Portais web	tailorability	Proprietário	PHP
XOOPS (www.xoops.org)	Portais web	tailorability	Proprietário	PHP
Zope (www.zope.org)	Portais web	tailorability	Proprietário	Python
DotNetNuke (www.dotnetnuke.com)	Portais web	tailorability	Proprietário	Asp.NET / VB.NET
ACOST [Hummes & Merialdo, 2000]	Ferramentas de comunicação síncrona	tailorability	Extensão do JavaBeans	Java
Flexible JAMM [Begole et al., 1999]	Applets colaborativos	linha de produto	Swing	Java
TOP [Guerrero & Fuller, 1999]	Aplicações colaborativas na Web	linha de produto	Proprietário	Java
MoCA [Sacramento et al., 2004]	Colaboração em dispositivos móveis	linha de produto	Proprietário	Java
Sieve [Isenhour et al., 1997]	Visualização colaborativa de dados	linha de produto	Extensão do JavaBeans	Java
Prospero [Dourish, 1998]	Groupware em geral	linha de produto	Proprietário	CLOS
COPSE-Web [David & Borges, 2004]	Groupware para Web	linha de produto	Proprietário	Java
TeleComputing Developer [Anderson et al. , 2002]	Groupware em geral	linha de produto	Proprietário	Java
Agilo [Guicking et al., 2005]	Groupware síncrono	linha de produto	Proprietário	Java

Tabela 2.1. Plataformas para o desenvolvimento de groupware baseado em componentes

As abordagens encontradas na literatura com relação à utilização de componentes de software no desenvolvimento de groupware se concentram basicamente na segunda e terceira motivação identificadas por Szyperski [2003] (linha de produto e *tailorability*), conforme ilustrado na Tabela 2.1. Os sistemas oferecem um ferramental para construir aplicações colaborativas a partir de componentes interoperáveis e, alguns deles, oferecem a possibilidade de composição pelos usuários-finais. Também pode ser notado na Tabela 2.1 que a maioria das plataformas analisadas utiliza a linguagem Java como base para sua implementação e não há um modelo de componentes padrão.

O foco da abordagem proposta nesta tese é na segunda motivação (linha de produto), visando instrumentar o desenvolvedor de groupware na montagem de sistemas colaborativos. Entretanto, alguma flexibilidade é oferecida ao usuário final, que seleciona e instala novos serviços de colaboração. Isto os possibilita, até certo ponto, adaptar a aplicação colaborativa para suas necessidades específicas e acompanhar as características do grupo e das tarefas.

A engenharia do domínio é propícia para utilização em domínios que apresentam processos e características complexos, onde há dificuldade de modelagem utilizando os processos tradicionais [Braga, 2000], que é o caso de CSCW e groupware. O modelo 3C instrumenta a análise do domínio e o desenvolvimento de groupware como um todo. O modelo 3C é tratado em mais detalhes no próximo capítulo.

3
O Modelo 3C de Colaboração

Para projetar groupware de qualidade, é necessário entender de colaboração. O objetivo deste capítulo é apresentar o conceito de colaboração e introduzir o modelo 3C. A colaboração, intrinsecamente complexa, é analisada para nortear o desenvolvimento de groupware.

Este capítulo segue a seguinte estrutura. O conceito de colaboração é definido e discutido na Seção 3.1 e o modelo 3C de colaboração é apresentado na Seção 3.2. Na Seção 3.3 são apresentados o ambiente AulaNet e o curso TIAE, que servem como estudo de caso para os conceitos apresentados na seqüência. Nas seções 3.4, 3.5 e 3.6 são abordadas respectivamente a comunicação, a coordenação e a cooperação. Na Seção 3.7 é apresentado um exemplo de classificação de uma ferramenta, na Seção 3.8 o modelo 3C é comparado a outras abordagens e modelos da literatura, e na Seção 3.9 são apresentadas as considerações finais.

3.1.
A Colaboração

O termo colaboração precisa ser contextualizado para definir a relação desejada entre os participantes [Brna, 1998]. Neste trabalho a colaboração é vista a partir da comunicação, coordenação e cooperação, conforme será discutido na próxima seção. Nesta seção são delineadas as características da colaboração, contextualizando o modelo 3C e o trabalho colaborativo. Colaboração é uma maneira de trabalhar em grupo, onde os membros do grupo atuam em conjunto visando o sucesso do projeto, sendo que a falha de um dos participantes normalmente implica na falha do grupo como um todo [Grosz, 1996]. Na colaboração, os participantes se ajudam objetivando o sucesso das tarefas e entre eles há uma hierarquia menos rígida, formando uma estrutura onde pares atuam

em conjunto com objetivos comuns e compartilhados. De acordo com Barros [1994]:

> colaborar (co-labore) significa trabalhar junto, que implica no conceito de objetivos compartilhados e uma intenção explicita de somar algo – criar alguma coisa nova ou diferente através da colaboração, se contrapondo a uma simples troca de informação ou de instruções.

Para evitar o fracasso do grupo na realização das tarefas interdependentes, os participantes planejam e agem em conjunto. Na colaboração, os participantes se empenham para o sucesso do grupo, o que favorece uma postura pró-ativa e participativa dos indivíduos e uma maior união do grupo. Normalmente a liderança muda durante a resolução das tarefas de acordo com as competências de cada um, de modo que os papéis se revezam entre os participantes, mesmo que um deles tenha mais poder e seja o responsável pela tarefa [Fielding, 1999].

Grosz [1996] exemplifica a colaboração na preparação de um jantar por um grupo. O objetivo é comum, de modo que se um falhar, o jantar falha. Desta forma, eles são compelidos a se ajudar. Quando um for ao supermercado, traz os ingredientes para os outros, e todos negociam o cardápio, para que seja coerente, e o uso dos recursos, para que um não utilize um recurso imprescindível para outro. A comunicação é voltada para ação, com objetivo de negociar e trocar informações. A coordenação é feita em duas etapas: na preparação para o trabalho e dinamicamente enquanto ele acontece. A cooperação acontece na realização conjunta das tarefas. Os indivíduos planejam juntos, atuam conjuntamente, negociam e possuem um objetivo compartilhado, sendo que a falha de um implica na falha de todos.

A colaboração é de grande valia no ambiente de trabalho, possibilitando ao grupo tratar tarefas complexas e que requerem habilidades multidisciplinares. No ambiente educacional, a colaboração também é valorizada e incentivada [Harasim et al., 1997]. Além da complementação de capacidades, do auxílio mútuo e da motivação advindos da colaboração, os novos profissionais são preparados a se relacionar, a negociar, a se expor, a liderar, a ter responsabilidade, e a se comunicar, coordenar e cooperar [Fuks, 2000].

A seguir, outros termos relacionados (interação, trabalho em grupo, competição e cooperação) são definidos e diferenciados do conceito de colaboração, visando sua clarificação e contextualização.

Interação é uma forma de relacionamento onde há trocas e influência mútua. Dirigir em uma grande cidade, por exemplo, é uma atividade interativa, porém normalmente não é colaborativa. Nesta atividade, não há um objetivo compartilhado pelo grupo, não há um comprometimento com o sucesso do outro e não há uma negociação sobre um plano compartilhado. Por outro lado, ao dirigir em comboio, os motoristas combinam o caminho, os checkpoints, se comunicam por sinais, rádio ou telefone, e se um precisar de ajuda, os outros param. O sucesso do grupo é todos chegarem ao destino. Ao dirigir em comboio, há um comprometimento com o sucesso dos companheiros e um objetivo comum e compartilhado, caracterizando a colaboração [Grosz, 1996].

Trabalho em grupo é um conjunto de atividades com objetivo de atingir um determinado fim, produzindo um resultado. No trabalho em grupo, não necessariamente o interesse do participante é atingir o objetivo do trabalho, pode ser, por exemplo, receber um pagamento, não ser castigado, etc. Para caracterizar a colaboração é necessário saber as intenções e objetivos dos participantes.

A *competição* se assemelha em muitos aspectos à colaboração. Ela é de natureza interativa e há um objetivo comum, porém conflitante. Ao invés dos indivíduos se ajudarem, eles disputam entre si os recursos e o sucesso de um normalmente implica no fracasso dos outros. Apesar disto, os concorrentes se comunicam (pouco), coordenam-se, seguindo regras normalmente pré-estabelecidas, e atuam em conjunto em um espaço compartilhado. Mesmo dentro da competição, em alguns casos os participantes colaboram. Por exemplo, nas cadeias de suprimento, os fabricantes de automóveis concorrentes se unem para definir padrões e fazer compras em conjunto, para cortar custos em aspectos comuns, sem prejudicar os diferenciais competitivos [Tapscoot et al., 2000].

Na literatura, é comum encontrar os termos colaboração e *cooperação* sendo usados indistintamente. Alguns pesquisadores diferenciam-nos de acordo com o grau de divisão do trabalho [Dillenbourg, 1999; Roschelle & Teasley, 1995; Brna, 1998]. Na cooperação, os membros do grupo executam tarefas individualmente e depois combinam os resultados parciais para obter o resultado final. Na

colaboração, os membros dos grupos trabalham juntos em um esforço coordenado [Dillenbourg & Self, 1992]. Brna [1998] classifica a colaboração como um estado e a cooperação como um dos processos necessários para estar no estudo de colaboração. No contexto deste trabalho, a cooperação é uma das atividades da colaboração.

3.2. O Modelo 3C de Colaboração

O modelo 3C de colaboração é baseado na concepção de que para colaborar, os membros de um grupo comunicam-se, coordenam-se e cooperam. O modelo 3C nasce do artigo seminal de Ellis et al. [1991]. O modelo de Ellis et al. é utilizado para classificação do suporte computacional à colaboração. Nesta tese, o modelo 3C é utilizado como base para a modelagem e desenvolvimento de groupware e cada C é profundamente analisado. Há também uma diferença de terminologia; a operação conjunta no espaço compartilhado é chamada por Ellis de colaboração, enquanto no modelo 3C é chamada de cooperação.

O modelo 3C é equivalente ao modelo Clover [Laurillau & Nigay, 2002]. Este modelo define três classes de funcionalidades: comunicação, coordenação e produção. O que é chamado de produção no modelo Clover corresponde ao conceito de cooperação no modelo 3C. Diferentemente do modelo Clover, nesta tese, o modelo 3C guia o desenvolvimento de groupware e dá origem a uma arquitetura componentizada.

Amiour [1997], Yang [1995], Amiour & Estublier [1998] e Bandinelli et al. [1996] utilizam as três dimensões do modelo 3C para aprimorar o suporte computacional a processos de software, principalmente nos aspectos da comunicação e cooperação, que, de acordo com eles, não são tratados adequadamente pelos processos tradicionais, que são mais voltados para a coordenação do grupo de desenvolvedores. Assim como Ellis et al. [1991], estes autores adotam definições de cooperação e colaboração invertidas em relação ao modelo 3C adotado nesta tese. Para eles, a cooperação é o trabalho em grupo, enquanto a colaboração é um dos aspectos da cooperação. Contudo, pela definição

apresentada por eles, pode-se afirmar que os modelos são equivalentes, apesar da inversão dos dois termos. De acordo com Amiour & Estublier [1998]:

> The taxonomy (…) identifies three main aspects of cooperation in software and business processes. The first one, Coordination, is the ordering of activities in the process; the second one, Collaboration, is related to the management of shared data; the last one, Communication, deals with the exchange of information between the performers of the process.

Assim como nesta tese, Amiour [1997] apresenta modelos específicos e uma maior elaboração para cada um dos Cs. Em sua proposta, o modelo de coordenação objetiva representar as atividades realizadas no trabalho em grupo através de regras capturadas em um modelo de eventos, diagramas de transição de estado e diagramas de fluxo de controle. O modelo de colaboração é voltado à representação dos produtos manipulados na cooperação e seu mapeamento e compartilhamento com as atividades realizadas. O modelo de comunicação está direcionado para a troca de mensagens, tratando as notificações e requisições, as respostas possíveis, o modo de entrega, bem com a forma de representação das mensagens.

A organização do modelo 3C também aparece freqüentemente na literatura em outros trabalhos, com outros enfoques. Bretain et al. [1997] utilizam os três Cs como base para analisar e entrevistar grupos cujas atividades são realizadas fora de um escritório, como bombeiros, encanadores, repórteres e representantes de vendas, objetivando definir um suporte computacional multimídia e móvel adequado às necessidades de cada grupo. Sauter et al. [1995] utilizam os três Cs para a classificação de groupware no escopo das empresas suíças. Sire et al. [1999] utilizam o modelo Clover para projetar o suporte computacional à coordenação em um groupware. Castellani et al. [1996] utilizam os três Cs para classificar ferramentas e para direcionar sua pesquisa. Magnussona & Svenssonb [2000] projetam o suporte computacional para grupos de estudantes utilizando os três Cs como base para classificação das ferramentas. Os três Cs são utilizados também por Muhammad et al. [2005], para o projeto do suporte à percepção em ambientes de produção de documentos web. Borghoff e Schlichter [2000] utilizam os três Cs para classificação de ferramentas colaborativas. Marsic & Dorohoceanu [2003] utilizam os três Cs para analisar elementos da interface com o usuário. Tatikonda & Stock [2003] utilizam os três Cs para analisar as relações

interorganizacionais no contexto da transferência de tecnologia em cadeias de suprimentos. Neale et al. [2004] utilizam os três Cs para avaliação de aplicações colaborativas. Teixeira & Chagas [2005] utilizam o modelo 3C para avaliação de ferramentas de co-autoria.

Figura 3.1. O diagrama do modelo 3C de colaboração

O diagrama do modelo 3C é apresentado na Figura 3.1. A comunicação envolve a troca de mensagens e a negociação de compromissos. Através da coordenação, as pessoas, as atividades e os recursos são gerenciados para lidar com conflitos e evitar a perda dos esforços de comunicação e de cooperação. A cooperação é a produção conjunta dos membros do grupo em um espaço compartilhado, gerando e manipulando objetos de cooperação na realização das tarefas. Apesar da separação destas atividades para fins de análise, a comunicação, a coordenação e a cooperação não são realizadas de maneira estanque e isolada; são realizadas continuamente e iterativamente durante o trabalho em grupo [Fuks et al., 2005]. As tarefas originam-se dos compromissos negociados durante a comunicação, são gerenciadas pela coordenação e são realizadas durante a cooperação. Através de mecanismos de percepção o indivíduo obtém *feedback* de suas ações e *feedthrough* das ações de seus colegas. Ao cooperar, é necessário renegociar e tomar decisões sobre situações inesperadas, o que requer novas rodadas de comunicação e coordenação.

A colaboração pode ser decomposta em atividades e cada atividade pode ser decomposta em subatividades com um planejamento, participantes e metodologias próprios. Cada uma destas subatividades possui necessidades distintas de comunicação, coordenação e cooperação. Antes de efetivamente executar uma tarefa, por exemplo, o grupo se organiza e se articula. Nesta atividade, também há necessidades específicas de colaboração, que são distintas das necessidades que

ocorrem durante a execução da tarefa. Os indivíduos que planejam podem não ser os mesmos que executam, como normalmente ocorre na linha de montagem, onde as atividades são planejadas e posteriormente cada indivíduo realiza sua tarefa sem interagir diretamente com os demais. Na colaboração, o plano é renegociado dinamicamente, não sendo possível separar plenamente a coordenação da cooperação. Enquanto os indivíduos colaboram, eles aprendem e refinam os processos de trabalho, renegociando os planos iniciais e intercalando ação e negociação. O groupware deve dar suporte a esta flexibilidade de renegociar os planos e exercer paralelamente a comunicação, coordenação e cooperação. Uma atividade específica de comunicação, como por exemplo, o bate-papo em um *chat*, requer comunicação (troca de mensagens), coordenação (políticas de acesso) e cooperação (registro e compartilhamento).

O ambiente de aprendizagem AulaNet e um de seus cursos são usados neste capítulo como estudos de caso para apresentar os conceitos do modelo 3C e nos capítulos seguintes como instanciação da abordagem proposta nesta tese. Na próxima seção, o ambiente e o curso são apresentados sucintamente.

3.3. O Ambiente AulaNet e o Curso TIAE

O AulaNet é um ambiente baseado em uma abordagem de groupware para o ensino-aprendizagem na web que vem sendo desenvolvido desde Junho de 1997 pelo Laboratório de Engenharia de Software da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). O AulaNet é gratuito e está disponível nas versões em português, inglês e espanhol em http://www.eduweb.com.br. O Projeto AulaNet recebeu menção honrosa no III Prêmio Alcatel à Inovação Tecnológica Brasil em 2000. Milhares de cópias do ambiente foram distribuídas e são utilizadas em diversas universidades e empresas.

O AulaNet oferece uma interface padronizada para participação em cursos através da web, conforme ilustrado na Figura 3.2. Esta interface é composta de uma janela principal e de um menu representado graficamente através de uma figura de controle remoto. A janela principal é por onde os aprendizes interagem com os conteúdos didáticos, com o mediador e com os demais aprendizes. O

controle remoto é um menu de serviços que fornece uma facilidade de navegação construída através da seleção prévia, feita pelo docente, dos serviços de comunicação, coordenação e cooperação.

Figura 3.2. A interface do ambiente AulaNet

Em cursos do AulaNet, um docente pode assumir três papéis: coordenador do curso, docente co-autor e mediador. O coordenador é o responsável pela estruturação do curso, selecionando quais serviços estarão disponíveis, configurando o espaço compartilhado e definindo a ementa, a metodologia, os conteúdos didáticos e outras informações sobre o curso. O coordenador conta com o auxílio de docentes co-autores responsáveis pela produção e inserção de conteúdos didáticos nos serviços selecionados. O mediador é quem cuida do dia-a-dia do curso e avalia a participação dos aprendizes.

No AulaNet é disponibilizado um conjunto de serviços e o coordenador seleciona os que serão utilizados em seu curso e configura-os de acordo com as dinâmicas educacionais que serão adotadas nas turmas. Nas suas primeiras versões, os serviços do AulaNet eram classificados em serviços *administrativos*, de *avaliação* e *didáticos*, que é uma abordagem comum em ferramentas educacionais [Edutools, 2005]. Entretanto, esta abordagem levou os docentes que

usavam o ambiente a ensinar da maneira vertical tradicional: professando informações com pouca interação entre eles e os aprendizes, e sem interação entre os aprendizes. Contudo, o que se espera de um aprendiz na colaboração é um alto grau de interação com seus colegas e com os docentes, que por sua vez devem agir como mediadores e coordenadores ao invés de entregadores de informação. Os serviços do AulaNet foram reorganizados com base no modelo 3C de colaboração, para incentivar a colaboração [Fuks, 2000].

Figura 3.3. Posicionamento dos serviços do AulaNet no triângulo apresentado por Borghoff & Schlichter [2000]

Os serviços de colaboração do ambiente AulaNet são atualmente organizados em serviços de comunicação, de coordenação e de cooperação. A Figura 3.3 ilustra o posicionamento dos serviços do AulaNet no triângulo apresentado em [Borghoff & Schlichter, 2000]. Os serviços do AulaNet estão posicionados na parte externa do triângulo. Os serviços são classificados de acordo com o seu propósito principal, conforme ilustrado na Figura 3.4.
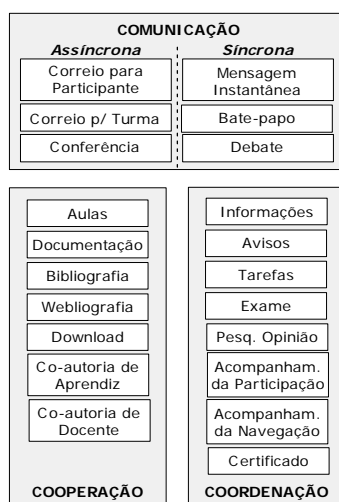
Figura 3.4. Classificação dos serviços do AulaNet com relação ao modelo 3C

Os serviços de comunicação objetivam a troca de informações, a argumentação e a negociação. Estes serviços incluem ferramentas de discussão textual assíncrona (*Conferências*), síncrona (*Debate*), de troca instantânea de mensagens (*Mensagem Instantânea*), e de correio eletrônico individual com o mediador e com toda a turma (*Correio para Participante* e *Correio para Turma*). Os serviços de coordenação visam o gerenciamento do grupo e incluem uma ferramenta de notificação (*Avisos*), ferramentas de avaliação (*Tarefas* e *Exames*) e uma ferramenta de acompanhamento da participação do grupo (*Relatórios de Participação*). Os serviços de cooperação do AulaNet incluem serviços de disponibilização de conteúdos (*Aulas*, *Documentação*, *Bibliografia* e *Webliografia*), transferência de conteúdo (*Download*) e mecanismos de co-autoria, tanto de docentes (*Co-autoria de Docente*) quanto de aprendizes (*Co-autoria de Aprendiz*).

O AulaNet pode ser utilizado para apoiar a sala de aula tradicional, apesar de ser mais propício para dar suporte à aprendizagem colaborativa. O curso TIAE (Tecnologias de Informação Aplicadas à Educação), que exemplifica este uso, é ministrado desde 1998 totalmente a distância pelo ambiente AulaNet como uma disciplina do Departamento de Informática da PUC-Rio, com o código INF 2133 para a pós-graduação do departamento e INF 1638 para a graduação. O objetivo do curso é que aprendizes colaborarem utilizando as tecnologias de informação, tornando-se educadores baseados na web [Fuks et al., 2002]. O curso visa construir uma rede de aprendizagem colaborativa onde o grupo aprende, primordialmente, através das interações entre os participantes. O curso também visa explorar a potencialidade da aprendizagem colaborativa, como fonte de inspiração para novos desenvolvimentos e para refinar o suporte computacional existente no AulaNet. O curso é coordenado pelos professores Carlos José Pereira de Lucena e Hugo Fuks e possui mediadores variados a cada semestre[7].

A metodologia do curso foi planejada para que, além de aprender os conteúdos do curso, os alunos habituados a serem receptores passivos se transformem em aprendizes geradores de conhecimento, aptos a trabalhar de forma colaborativa. No curso, o aprendiz é levado a aprender a buscar suas

[7] Fui mediador do segundo semestre de 2000 ao primeiro semestre de 2004.

próprias fontes de informação, a lidar com a sobrecarga e a converter colaborativamente informação em conhecimento. Os aprendizes tornam-se responsáveis pelo sucesso da aprendizagem ao gerarem conteúdos didáticos, argumentarem, dinamizarem as discussões e contribuírem com o aprendizado dos colegas. Eles são avaliados pelas contribuições que agregam valor ao grupo e não somente por suas atividades individuais [Fuks et al., 2003]. Alguns pontos da dinâmica do curso são caracterizados a seguir para clarificar o estudo de caso.

Na primeira parte do curso, um tópico é abordado a cada semana, durante oito semanas. A seqüência de atividades desta fase do curso é apresentada na Figura 3.5. Os aprendizes lêem os conteúdos selecionados sobre o tópico, realizam pesquisas de aprofundamento e participam de uma discussão sobre questões específicas sobre o tópico em estudo.

Sexta	Sábado	Domingo	Segunda	Terça	Quarta	Quinta
Estudo (Aulas, Documentação e pesquisas Web)						
			Seminário (Conferência) 12hs		14hs	
						Debate 13 às 14hs

Figura 3.5. Seqüência de atividades durante o estudo dos tópicos do curso

A discussão sobre cada um dos tópicos do curso é realizada durante 50 horas através do serviço Conferências do AulaNet, que funciona como um fórum de discussão, onde é possível encadear e categorizar as mensagens (Figura 3.6a) [Gerosa et al., 2001]. Após a discussão na conferência, o tópico em estudo é encerrado com a realização de um debate síncrono, com duração de uma hora, pela ferramenta Debate do AulaNet (Figura 3.6b).
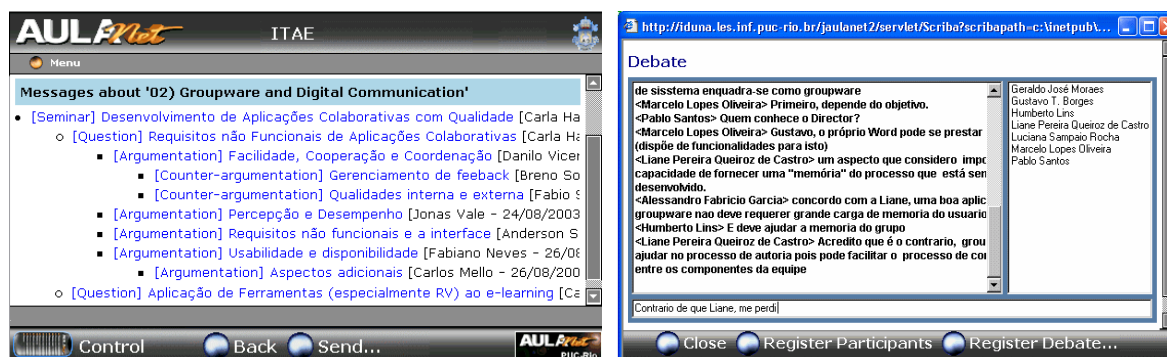
Figura 3.6. Trecho de diálogo na Conferência (a) e no Debate (b)

Na segunda parte do curso, os aprendizes desenvolvem em grupo um conteúdo educacional multimídia e interativo. Após a entrega, é realizada uma

revisão pelos pares, onde membros de pelos menos outros três grupos avaliam cada conteúdo produzido. Esta avaliação acontece em conferências criadas especificamente com este propósito, onde os aprendizes discutem os problemas encontrados nos protótipos de conteúdos. Ao final deste período, os grupos têm um prazo para apresentar a versão revisada que incorpora as contribuições de seus colegas. A Tabela 3.1 apresenta o cronograma da edição 2005.2 do curso.

Etapa	Atividade	Data e horário	Serviço
Apresentação	Aula presencial inaugural	11/08 12hs - 13hs	sala de aula
	Apresentar-se para a turma	12/08 - 15/08	Correio para Turma
	Preenchimento do Perfil	12/08 - 15/08	Configurar Perfil
Estudo e discussão dos tópicos do curso	1) Introdução ao AulaNet e ao curso TIAE	12/03 - 18/03	Aulas
		15/08 12hs - 17/08 14hs	Conferências
		18/08 12hs - 13hs	Debate
	2) Groupware e comunicação digital	19/08 - 25/08	Aulas
		22/08 12hs - 24/08 14hs	Conferências
		25/08 12hs - 13hs	Debate
	3) Instrução baseada na Web (IBW) e a sala de aula tradicional	26/08 - 01/09	Aulas
		29/08 12hs - 31/08 14hs	Conferências
		01/09 12hs - 13hs	Debate
	4) Learningware e ambientes para IBW	02/09 - 15/09	Aulas
		12/09 12hs - 14/09 14hs	Conferências
		15/09 12hs - 13hs	Debate
	5) Papel do facilitador em IBW e conceitos sobre aprendizagem	16/09 - 22/09	Aulas
		19/09 12hs - 21/09 14hs	Conferências
		22/09 12hs - 13hs	Debate
	6) Ensinando, aprendendo e implantando IBW	23/09 - 29/09	Aulas
		26/09 12hs - 28/09 14hs	Conferências
		29/09 12hs - 13hs	Debate
	7) Multimídia interativa e design de cursos para IBW	30/09 - 06/10	Aulas
		03/10 12hs - 05/10 14hs	Conferências
		06/10 12hs - 13hs	Debate
	8) Novos rumos de IBW	07/10 - 20/10	Aulas
		17/10 12hs - 19/10 14hs	Conferências
		20/10 12hs - 13hs	Debate
Produção de conteúdo interativo hipermídia	Produção e submissão da versão-protótipo	21/10 - 03/11	Tarefas
	Avaliação colaborativa da versão-protótipo	04/11 12hs - 11/11 12hs	Conferências
	Produção e submissão da versão-final	11/11 - 24/11	Tarefas
	Avaliação colaborativa da versão-final	25/11 - 28/11	Correio para Participantes
Encerramento	Entrega das notas	30/11	Correio para Turma
	Prova Final (somente para os que não obtiveram nota mínima)	01/12 12hs - 13hs	sala de aula

Tabela 3.1. Cronograma de atividades do curso TIAE em 2005.2

O curso TIAE visa desenvolver a capacidade de trabalho na sociedade conectada, aumentando o grau de interação entre os aprendizes, exercitando suas capacidades de comunicação, coordenação e cooperação. São também capacitados a gerarem conhecimento de forma colaborativa, selecionando e filtrando conjuntamente a massa de informações disponível. O curso e o ambiente AulaNet

são utilizados nas próximas seções como estudos de caso para a comunicação, coordenação e cooperação.

3.4.
Comunicação: Argumentação para Ação

De acordo com o dicionário Houaiss [2001], comunicação é o:

> processo que envolve a transmissão e a recepção de mensagens entre uma fonte emissora e um destinatário receptor, no qual as informações, transmitidas por intermédio de recursos físicos (fala, audição, visão etc.) ou de aparelhos e dispositivos técnicos, são codificadas na fonte e decodificadas no destino com o uso de sistemas convencionados de signos ou símbolos sonoros, escritos, iconográficos, gestuais etc.

No trabalho em grupo a comunicação é principalmente voltada para ação [Winograd & Flores, 1987]. Quando o trabalho está todo pré-articulado, a comunicação é verticalizada: as ordens descem a hierarquia e os relatórios sobem; a comunicação horizontal, com o colega ao lado, além de não ser bem vista não tem suporte tecnológico. Na colaboração, os pares interagem argumentando e negociando compromissos. Através da comunicação, o grupo debate pontos de vista para alinhar e refinar as idéias, o que é fundamental para que o grupo consiga realizar tarefas interdependentes, não completamente descritas ou que necessitem de negociação [Fussel et al., 1998]. Alguns exemplos de ferramentas de comunicação atualmente utilizadas são: e-mail, lista de discussão, fórum, ferramentas de CSCA (*Computer Supported Collaborative Argumentation*), mensagem instantânea, chat, vídeo-conferência, teleconferência, telefone, etc. [Long & Baecker, 1997].

Ao se comunicar, um dos interlocutores, de acordo com suas intenções e compromissos, formula a mensagem a ser transmitida, e o outro, ao receber e interpretar a mensagem, tem seus compromissos modificados. Os interlocutores negociam suas intenções e compromissos através de mensagens formuladas através de signos, com significantes e significados [Blikstein, 2000].

Para viabilizar a comunicação, a linguagem utilizada na conversação deve ser entendida por todos os envolvidos. A linguagem é influenciada pelo contexto cultural, pelo domínio em questão, pelos conhecimentos individuais dos envolvidos e pelos recursos disponíveis para conversação. A linguagem define o

código que associa significantes e significados [Blikstein, 2000]. O emissor formula sua mensagem, codificando-a em signos, expressa-a para a ferramenta de comunicação, que a captura, transmite e apresenta ao receptor em mecanismos de percepção para que seja interpretada [Gerosa et al., 2003]. O ambiente define o espaço compartilhado de informações entre os indivíduos e fornece elementos não-verbais, como gestos, cores e expressões faciais, à linguagem utilizada na conversação [Gutwin & Greenberg, 2002].

Quando se comunicam, os interlocutores normalmente se concentram na argumentação, utilizando sem se dar conta a linguagem, os mecanismos de expressão e de percepção e a infra-estrutura do canal de dados. Porém, se for detectado algum tipo de confusão ou problema, a linguagem, o registro e o canal são trazidos para o foco central, em uma tentativa de encontrar o motivo do desentendimento. Para haver entendimento e a comunicação cumprir o seu objetivo, além do conhecimento da linguagem, é necessária a utilização adequada das mídias de transmissão e de recebimento dos dados, bem como a participação ativa do receptor, que deve estar atento às informações transmitidas e aos elementos utilizados.

Em uma comunicação bem sucedida, o conteúdo recebido é semanticamente equivalente ao transmitido. A única forma de se obter indícios do sucesso da comunicação é através do discurso e das ações (e reações) do receptor, pois são guiadas por seus compromissos e conhecimentos. Uma ruptura na comunicação causa uma discordância entre as intenções do emissor e as ações do receptor ao realizar os compromissos. A ruptura pode ser decorrente de uma falha em qualquer ponto da comunicação, desde a concepção e formulação da mensagem até sua interpretação e entendimento. As falhas são provenientes de fatores como desconhecimento das regras ou da linguagem, ruídos, interrupções, excesso de informação, falta de atenção, ambigüidade, conflitos, etc. [Blikstein, 2000].

Para projetar uma ferramenta de comunicação devem ser considerados diversos elementos da comunicação. O projetista da ferramenta define o que é relevante para sua ferramenta, de acordo com as necessidades de comunicação esperadas, considerando tempo, espaço, propósito, dinâmica e tipo de participante [Fuks et al., 2003]. A maneira como os interlocutores se comunicam é

influenciada pelos recursos e características da ferramenta, de modo que, após a separação em partes, o projeto da ferramenta é considerado como um todo.

A captura, a transmissão e a apresentação das mensagens constituem a base da comunicação eletrônica, tendo suporte em todas as ferramentas de comunicação. A mídia, o modo de transmissão, o tipo de comunicação e as restrições do canal influenciam diretamente estas atividades. A mídia pode ser textual, falada, gesticulada, com o uso de avatar ou de vídeo, ou pictórica, com imagens ou *emoticons*. Quanto mais rica for a mídia utilizada, maior a expressividade [Daft & Lengel, 1986]. No caso de uma videoconferência, por exemplo, obtém-se diversas informações pela linguagem corporal, pelas expressões faciais, pela entonação de voz, etc.

O modo de transmissão pode ser contínuo ou em blocos. No modo contínuo, são transmitidos pacotes, que individualmente não caracterizam uma mensagem. Por exemplo, em uma videoconferência a mensagem é diluída na transmissão contínua de informações por áudio e vídeo, de forma verbal e não-verbal. Em um chat onde cada participante enxerga simultaneamente o que os outros estão escrevendo, um interlocutor começa a responder antes mesmo que o outro termine a frase, entrelaçando o discurso. O modo de transmissão contínuo faz sentido na comunicação síncrona, onde os interlocutores estão simultaneamente conectados. Já no modo em blocos, o emissor prepara um pacote de informações (a mensagem) e depois o envia ao receptor. O emissor possui um espaço privativo onde trabalha no conteúdo antes de se expor. O modo de transmissão em blocos é usado para a comunicação síncrona ou assíncrona. Normalmente, ferramentas de comunicação assíncrona são utilizadas quando se deseja valorizar a reflexão por parte dos interlocutores, visto que terão mais tempo antes de agir [Benbunan-Fich & Hiltz, 1999]. Nas ferramentas de comunicação síncrona, a velocidade de interação é mais valorizada, dado o baixo tempo de latência entre as ações dos interlocutores. O tipo de comunicação esperado modifica o tratamento dado às mensagens em termos de exibição e de transmissão.

Eventualmente, são definidas restrições ao canal de comunicação. Por exemplo, é possível restringir o tamanho do texto, caracteres permitidos, taxa de transmissão (para áudio e vídeo) e latência. Estas restrições são utilizadas nas ferramentas para reduzir a sobrecarga de informação ou para reduzir o volume de

dados a transmitir. O suporte computacional oferecido para definir ou restringir a linguagem estabelecida na conversação está relacionado à restrição ou ampliação do vocabulário disponível ou ao estabelecimento de meta-informações sobre a mensagem. Ao situar o conteúdo, torna-se a comunicação mais sucinta, pois parte do que precisaria ser dito passa a ser inferido pelo contexto. Muitas vezes, as meta-informações explicitam informações que estariam implícitas no discurso. Meta-informações comumente encontradas em ferramentas de comunicação são o assunto ou título da mensagem, a data, a prioridade e a categoria. A categoria, em especial, pode ser utilizada para complementar a semântica da mensagem. O emissor seleciona uma categoria de um conjunto pré-definido e os receptores contextualizam a interpretação da mensagem. A categorização é utilizada também para direcionar uma dinâmica de diálogo, detectar conflitos, identificar a resolução de tarefas e organizar informações [Gerosa et al., 2001].

A estruturação da conversação também é levada em consideração no projeto da ferramenta, que pode ser voltada para uma conversa estruturada em lista, em árvore ou em grafo, conforme ilustrado na Figura 3.7 [Fuks et al., 2003]. A estruturação explicita visualmente as inter-relações entre as mensagens, que normalmente ficam implícitas no texto.
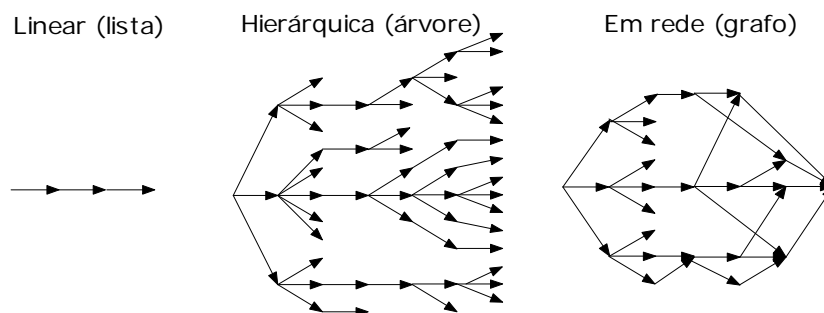
Figura 3.7. Exemplos de estruturação da discussão

O tipo de estruturação satisfaz as demandas de comunicação do grupo, enquanto ele argumenta para ação. Apesar de a lista ser um caso particular da árvore, e esta ser um caso particular do grafo, nenhuma das estruturas é sempre melhor do que as outras. A estruturação linear é propícia quando a ordem cronológica é mais importante do que as eventuais relações entre as mensagens, como no envio de avisos, informes e notícias. A estruturação hierárquica é propícia para a visualização da largura e da profundidade da discussão, possibilitando o encadeamento de mensagens sobre o mesmo assunto em um

mesmo ramo. Porém, como não há como ligar uma mensagem de uma ramificação a outra, a árvore só pode crescer, de modo que a discussão ocorre em linhas divergentes [Stahl, 2001]. A estruturação em rede (grafo) é utilizada para buscar convergência da discussão [Kirschner et al., 2003].

Outros fatores considerados ao projetar a ferramenta, com relação à linguagem, são o vocabulário permitido e o proibido, as maneiras de chamar a atenção, as regras de construção, as seqüências legais e os caminhos de conversação. Um exemplo de ferramenta com caminhos de conversação, que restringem as possíveis direções que a conversação pode tomar, é o Coordinator [Winograd & Flores, 1987]. Nesta ferramenta, a conversação só pode fluir por caminhos pré-definidos. Este recurso formaliza a conversação e não é usado quando a fluência é requerida.

A argumentação está ligada à semântica da conversação. Para oferecer suporte computacional à argumentação, a ferramenta atua na detecção e correção de rupturas, na estruturação da argumentação, no suporte a dialética e retórica ou na gestão dos compromissos negociados. Neste último caso, pode-se utilizar o modelo descrito por [Mackenzie, 1985] e [Raposo et al., 2004] para oferecer suporte computacional ao processo de argumentação e negociação. Para utilizar estes modelos, a conversação deve ser formal e seguir uma máquina de estados bem definida [Laufer & Fuks, 1995].

Por lidar com conhecimentos, intenções e compromissos, oferecer suporte computacional à argumentação não é trivial. A ferramenta deve oferecer suporte direto a atividades como negociação, persuasão, convencimento, entendimento, alinhamento de idéias, entre outras. Muitas vezes, o projetista necessita restringir a capacidade de comunicação, para evitar ambigüidades e inconsistências e possibilitar a interpretação mais precisa do discurso.

3.4.1.
Estudo de Caso da Comunicação no AulaNet e no Curso TIAE

Nesta seção, a comunicação no curso TIAE é analisada. A comunicação tem um papel fundamental no processo de ensino-aprendizagem, possibilitando a troca

de informações e pontos de vista, além de interconectar o grupo. No curso TIAE são utilizados todos os serviços de comunicação do ambiente AulaNet.

Durante a argumentação, os aprendizes atacam e defendem os pontos de vista e conceitos apresentados, buscando, estruturando e validando informações [Kanselaar et al., 2003]. No TIAE, as idéias, os pontos de vistas e os argumentos são expostos e entendidos, sem necessariamente alcançar uma única solução para as questões ou chegar a um acordo ou consenso. No TIAE, é valorizada a argumentação gerada do confronto das idéias diferentes, pois é esperado que o aprendizado decorra desta argumentação e do respectivo alinhamento de idéias e não da harmonização e do consenso. A argumentação que ocorre durante a primeira fase do curso é realizada em duas etapas: discussão assíncrona nas Conferências e discussão síncrona no Debate.

Nas atividades assíncronas, os aprendizes participam em um horário e local mais conveniente e apropriado para a tarefa. A organização, estruturação do pensamento, reflexão e aprofundamento da discussão são favorecidos [Funaro & Montell, 1999; Benbunan-Fich & Hiltz, 1999]. As conferências funcionam no estilo de fórum, sendo possível postar mensagens respondendo, comentando ou criticando outra mensagem, criando um encadeamento na exibição das mensagens. A Figura 3.8 ilustra um trecho de diálogo em uma conferência.
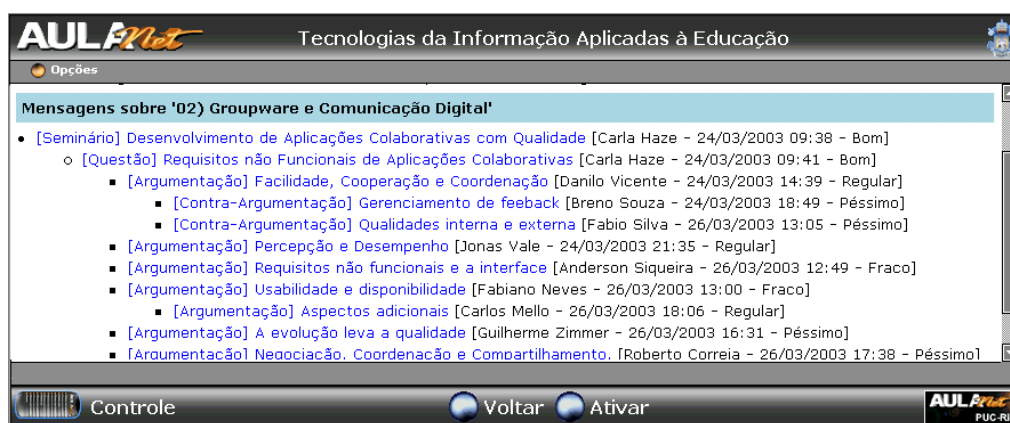
Figura 3.8. Trecho de um diálogo em uma Conferência

A comunicação no curso TIAE é predominantemente textual. A pluralidade de mídias é tratada na última parte do curso, quando os aprendizes produzem conteúdos educacionais multimídia e interativos. Nesta etapa, o serviço Conferências é utilizado para revisão em pares dos protótipos de conteúdos desenvolvidos por cada grupo. Os aprendizes negociam as revisões que são

necessárias para os protótipos e cada grupo prepara a versão final, levando em consideração os compromissos assumidos durante a negociação.

As ferramentas do TIAE adotam o modo de transmissão em blocos, que além de favorecer a reflexão, é mais propício para grupos numerosos [Long & Baecker, 1997]. No Debate, é utilizada uma estruturação linear e são adotadas algumas restrições no canal de comunicação, como por exemplo, limite de tamanho da mensagem, com o objetivo de reduzir a confusão na conversação [Pimentel et al., 2005]. O serviço Conferências utiliza uma estruturação hierárquica, que propicia a organização do discurso e provê indicações sobre a evolução da turma, auxiliando na identificação de discussões que diferem do esperado [Gerosa et al., 2005]. Nas árvores apresentadas na Figura 3.9 é possível observar a interação declinando ao longo do tempo na turma de 2002.1. Nas primeiras quatro conferências, a profundidade média das árvores foi 3,0 e a porcentagem de mensagens não respondidas (folhas) foi 51%. Nas quatro últimas conferências, a profundidade foi 2,8 e a quantidade de folhas foi 61%. Neste caso a estrutura da conversação poderia ter sido utilizada para identificar a redução da interação na turma, indicada pela baixa profundidade das árvores e alta quantidade de folha.
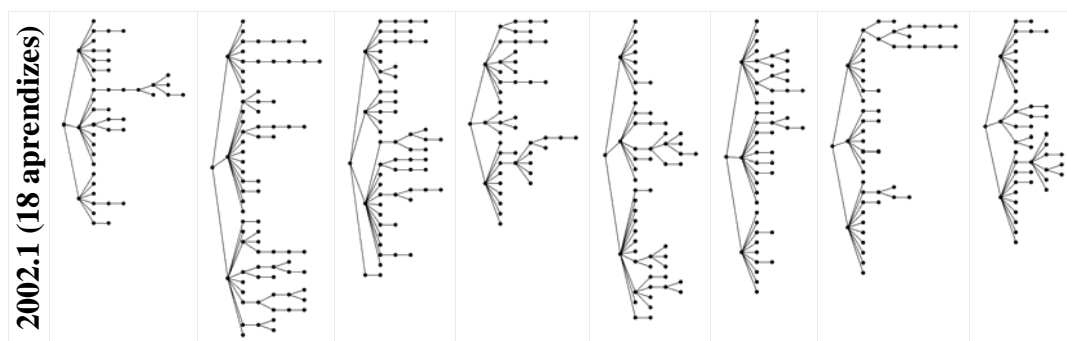
Figura 3.9. Árvores derivadas das conferências de uma edição do curso TIAE

O serviço Conferências utiliza como meta-informações o assunto, a data e a categoria da mensagem. O autor escolhe a categoria mais apropriada para sua mensagem, provendo semântica para os relacionamentos. As categorias adotadas no TIAE foram originalmente baseadas nos tipos de nós do IBIS [Conklin & Begeman, 1988]. Atualmente, as categorias em uso são: Seminário, Questão, Argumentação, Contra-Argumentação e Esclarecimento. A Figura 3.10 apresenta um trecho de um diálogo de uma conferência ilustrando o mapeamento das mensagens numeradas para uma árvore categorizada.
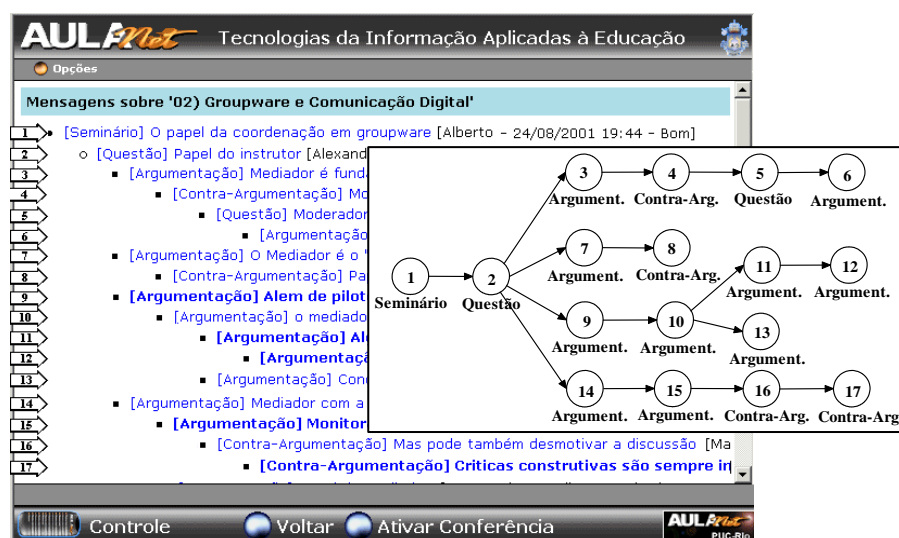
Figura 3.10. Árvore derivada de uma conferência
ressaltando as categorias das mensagens

As categorias auxiliam no entendimento do relacionamento entre as mensagens, complementando a informação provida pela estruturação hierárquica e auxiliando na identificação da direção que a discussão está tomando [Gerosa et al., 2005]. Por exemplo, em uma árvore ou ramo que só contém mensagens de argumentação, não está havendo confronto de idéias, enquanto um número excessivo de contra-argumentações indica que o grupo está discutindo um único assunto ou se evolveu em um conflito interpessoal.

A Tabela 3.2 sumariza o mapeamento dos elementos de comunicação utilizados nos serviços de comunicação do AulaNet.

Elemento	**Correio para Participante**	**Correio para Turma**	**Conferências**	**Debate**	**Mensagem Instantânea**
Mídia	Textual	Textual	Textual	Textual	Textual
Modo de Transmissão	Bloco	Bloco	Bloco	Bloco	Bloco
Estrutura da conversação	Hierárquica	Linear	Hierárquica	Linear	Linear
Meta-informação	Assunto, data	Assunto, data	Assunto, data	Hora	-
Categoria	-	Disponível	Disponível	-	-
Restrições	-	-	-	Tamanho	-

Tabela 3.2. Elementos de comunicação adotados nos
serviços de comunicação do AulaNet

Eventualmente, um serviço de comunicação também é utilizado com propósitos de coordenação ou cooperação. Os serviços Correio para Participante, Correio para Turma e Mensagem Instantânea são utilizados no TIAE com propósito de coordenação. Os moderadores enviam avisos, informes e alertas para organizar a participação dos aprendizes através destes serviços. Além disto,

mesmo um serviço de comunicação, necessita de elementos de coordenação e de cooperação, visto que a própria comunicação é uma atividade colaborativa. O suporte à coordenação na comunicação está relacionado principalmente com as políticas de acesso ao canal, e o suporte à cooperação com o registro e manipulação das informações.

3.5.
Coordenação: Gerenciamento de Interdependências

De acordo com o dicionário Houaiss [2001], coordenar é:

> organizar(-se) de forma metódica, estruturar, ordenar(-se); conjugar, concatenar, interligar; manter ou tornar sincrônico e harmonioso; ser responsável pelo andamento, pelo progresso de (setor, equipe, projeto etc.), dirigir; fazer combinação ou ajuste (de), acertar(-se).

No trabalho em grupo, a coordenação de atividades é necessária para garantir o cumprimento dos compromissos assumidos na comunicação e a realização do trabalho colaborativo através da soma dos trabalhos individuais. A coordenação organiza o grupo para evitar que esforços de comunicação e de cooperação sejam perdidos e para garantir que as tarefas sejam realizadas da forma mais adequada, no tempo certo e com os recursos necessários [Raposo & Fuks, 2002].

A coordenação de uma atividade envolve a pré-articulação de tarefas, o gerenciamento do seu andamento e a pós-articulação. A pré-articulação envolve a negociação necessária para preparar a colaboração, normalmente concluída antes do trabalho colaborativo ser iniciado: identificação dos objetivos, mapeamento dos objetivos em tarefas, seleção dos participantes, distribuição de responsabilidades, etc. Ao pré-articular as tarefas, a negociação ocorre a priori e, ao colaborar, o grupo segue o roteiro pré-estabelecido. Um exemplo de trabalho normalmente todo pré-articulado ocorre na linha de montagem, onde os participantes realizam suas tarefas individualmente seguindo o que foi estabelecido, sem necessidade de renegociar o trabalho. A completa pré-articulação das tarefas é apropriada quando a atividade e seu processo de resolução forem bem conhecidos. A pós-articulação ocorre após o término das

tarefas, e envolve a avaliação, análise e documentação do processo de colaboração.

Quando não é possível pré-articular totalmente as tarefas ou quando é necessário adaptar o planejamento, a coordenação é tratada dinamicamente durante o gerenciamento do andamento das tarefas e de suas interdependências. Esta coordenação, definida como “o ato de gerenciar interdependências entre as atividades realizadas para se atingir um objetivo” [Malone & Crowston, 1990], é renegociada de maneira quase contínua ao longo de todo o tempo. Apesar da interdependência ser normalmente positiva (um participante desejando que o trabalho do outro seja bem sucedido), ela nem sempre é harmoniosa. Sem coordenação, há o risco de os participantes se envolverem em tarefas conflitantes ou repetitivas [Raposo & Fuks, 2002].

Um coordenador pode ser definido para organizar o grupo durante a colaboração, reduzindo a necessidade de pré-articulação. O coordenador organiza os participantes, tarefas e recursos e acompanha a evolução dos processos de trabalho. Alguns grupos operam bem sem a presença de um coordenador explícito; os participantes se organizam dinamicamente enquanto a colaboração ocorre, ajustando-se dinamicamente às mudanças nas tarefas e ao seu entendimento [Dron et al., 2001]. Esta abordagem é apropriada para grupos pequenos, coesos e com participantes competentes e comprometidos [Teles, 2004]. Muitas vezes é utilizada uma abordagem híbrida, onde parte das tarefas são pré-articuladas, um coordenador é eleito para acompanhar o grupo e, em momentos específicos, os próprios participantes se coordenam. O grau de flexibilidade a ser adotado na coordenação depende dos participantes, das tarefas e dos recursos disponíveis. Ao distribuir a coordenação no grupo, reduz-se a dependência do coordenador, liberando-o para tratar da organização do grupo em um alto nível, bem como planejar ações futuras [Durfee, 1988].

Para o coordenador e para os participantes se coordenarem são necessárias informações de percepção que dêem ciência do que está acontecendo e do que as outras pessoas estão fazendo [Borges & Pino, 1999]. É importante que cada um conheça o progresso do trabalho dos companheiros: o que foi feito, como foi feito, o que falta para o término, quais são as mudanças de planos, as necessidades e características de cada um, o progresso das tarefas e os resultados preliminares.

Sem esta percepção mútua, ocorrem conflitos e duplicações desnecessárias de esforços [Dourish & Belloti, 1992].

O gerenciamento da percepção é, portanto, um elemento de coordenação considerado no projeto do sistema. São levadas em conta as informações de percepção referentes aos efeitos das ações do indivíduo (*feedback*) e de seus colegas (*feedthrough*). O fluxo de informações de percepção deve ser cuidadosamente planejado, já que, a princípio, toda informação sobre o que acontece, aconteceu ou acontecerá no grupo têm alguma importância, e o excesso de informações dificulta a tomada de decisões [Hwang & Lin, 1999]. Deve-se projetar também com cautela as informações de *feedthrough*, para não restringir excessivamente o espaço individual e a privacidade dos participantes [Gandon & Sadeh, 2004].

Na maioria dos bate-papos e videoconferências, o grupo coordena-se exclusivamente com base na percepção. Nestes casos, a coordenação fica a cargo do protocolo social, caracterizado pela ausência de mecanismos de coordenação explícitos entre as atividades. A coordenação nestas situações é estabelecida culturalmente [Gutwin & Greenberg, 2002]. Por outro lado, atividades cujas tarefas são altamente interdependentes não são satisfatoriamente coordenadas somente com o protocolo social. Nestes casos, são utilizados mecanismos de coordenação por software. Um mecanismo de coordenação é um dispositivo voltado a dar suporte ao trabalho de articulação [Schmidt & Simone, 1996]. Exemplos de ferramentas com mecanismos de coordenação explícitos são os gerenciadores de fluxo de trabalho (*workflow*), jogos multi-usuário e ferramentas colaborativas de autoria e de desenvolvimento de software.

Nem sempre é claro o que deve ficar a cargo do protocolo social e o que deve ter um mecanismo de coordenação associado. Em alguns casos a coordenação é melhor conduzida em função do *feedthrough*, em função de mecanismos através dos quais os participantes explicitamente sinalizam suas intenções e necessidades ou através de mecanismos de coordenação. Os mecanismos de coordenação propostos devem ser suficientemente flexíveis, dado o dinamismo da interação entre os participantes, e deve haver mecanismos que possibilitem aos usuários interpretar os padrões de trabalho, usá-los, modificá-los ou rejeitá-los [Schmidt, 1991].

Para coordenar é necessária uma definição clara de tarefas, atividades colaborativas e interdependências. No modelo adotado neste trabalho, uma atividade colaborativa é um conjunto de tarefas realizadas para se atingir um objetivo comum [Raposo & Fuks, 2002]. Tarefas compõem as atividades colaborativas e estão ligadas por interdependências, podendo ser atômicas ou compostas de subtarefas. Um grupo de subtarefas é considerado uma tarefa em um nível de abstração mais alto quando não apresenta interdependência com tarefas externas a este grupo.

A modelagem das tarefas e seus relacionamentos é outro elemento a ser considerado no projeto do suporte computacional à coordenação. É possível caracterizar diferentes tipos de interdependências e identificar mecanismos de coordenação para gerenciá-las [Malone & Crowston, 1994]. As interdependências são relacionadas ao tempo ou aos objetos de cooperação (recursos) [Ellis & Wainer, 1994]. No nível temporal a coordenação lida com o seqüenciamento das tarefas, enquanto no nível de objetos, a coordenação lida com o compartilhamento e com a concorrência de acesso.

No modelo proposto por Raposo & Fuks [2002], as interdependências estendem as relações temporais definidas por [Allen, 1984], adicionando operadores que estabelecem a semântica dos relacionamentos entre as tarefas. Para diferenciar entre a interpretação ativa e passiva de uma interdependência, dois operadores foram definidos: *enables* e *forces*. Com o operador *forces* é possível definir que o início ou fim da tarefa A força o início ou fim da tarefa B. Outros operadores definidos foram *blocks*, quando o início ou fim de uma tarefa bloqueia a outra, e *unblocks*, para a situação inversa. Este modelo de representação das tarefas é utilizado para criar mecanismos de coordenação que gerenciam as interdependências entre as tarefas, como o exemplo que utiliza redes de Petri apresentado por Raposo & Fuks [2002].

Na coordenação também são considerados o acompanhamento, a estruturação e a organização do grupo. O acompanhamento da participação é necessário para o coordenador monitorar o andamento das atividades de modo a intervir quando julgar necessário e para os participantes compararem e ajustarem seus esforços. Além da avaliação quantitativa, algumas vezes é necessária uma avaliação qualitativa [Fuks et al., 2003]. A avaliação qualitativa possibilita

identificar participações desnecessárias ou fora dos padrões. A avaliação das atividades também oferece subsídio para a gestão de competências dos participantes, que é influenciada pela qualificação, interesse e performance. A performance é extraída a partir da qualidade da participação nas tarefas colaborativas [Mitchell et al., 2004]. A gestão de competências é um elemento importante da coordenação por embasar decisões a respeito da formação de subgrupos, alocação de tarefas, capacitação do grupo, entre outros.

A estruturação do grupo inclui a definição dos papéis, da hierarquia, dos subgrupos e das permissões dos participantes. Estes elementos costumam ser dinâmicos e o groupware deve prover flexibilidade para se adaptar aos diferentes momentos da colaboração, onde os participantes assumem diferentes papéis e estruturas. No gerenciamento de permissões leva-se em conta o indivíduo e o papel que exerce no grupo. Ações comuns no gerenciamento de sessões incluem o convite e aceitação dos participantes, a definição do início e fim, do tema e da ordem de participação. Na organização do grupo, as tarefas são atribuídas aos participantes e é definida a dinâmica da colaboração.

3.5.1.
Estudo de Caso da Coordenação no AulaNet e no Curso TIAE

O AulaNet oferece dois papéis pré-definidos voltados à coordenação: coordenador e mediador. O coordenador cuida da pré-articulação, definindo as tarefas, os conteúdos e a dinâmica do curso. O coordenador cuida também da pós-articulação, avaliando e refinando o curso com base na realimentação obtida das edições anteriores [Gerosa et al., 2002]. O mediador lida com o gerenciamento das tarefas e com o dia-a-dia do curso [Salmon, 2000]. No curso TIAE, para deixar os próprios aprendizes se organizarem e se coordenarem, só há intervenção dos mediadores quando for realmente necessário. Desta maneira, os aprendizes se capacitam a trabalhar em grupo, e torna-se menos necessária a presença e atenção constante do mediador. Foram criados dois papéis para os aprendizes: seminarista da Conferência e moderador do Debate. Os aprendizes se revezam nestes papéis ao longo dos temas do curso.

O seminarista é responsável por iniciar a conferência enviando uma mensagem da categoria Seminário, onde aborda um aspecto do tema da semana, deixando clara sua intenção. Além desta mensagem, coloca três mensagens da categoria Questão a partir das quais a turma desenvolve a argumentação ao longo da semana. Durante este período de argumentação, o seminarista anima e mantém a dinâmica da conferência. Nos debates do curso, um aprendiz desempenha o papel de moderador, tornando-se responsável por conduzir a sessão, manter o foco nas questões propostas, manter o ritmo da discussão e coordenar os outros aprendizes, estimulando a participação de todos. Os demais aprendizes participam da discussão, argumentando seus pontos de vista, trabalhando ativamente seus conceitos, refletindo sobre os mesmos e refinando-os [Schön, 1983]. O trabalho dos aprendizes é observado, comentado e avaliado por seus colegas, motivando-os a participar com melhor qualidade [Benbunan-Fich & Hiltz, 1999].

A Figura 3.11 exibe a seqüência de atividades de todo o curso TIAE. O curso inicia-se com a apresentação da dinâmica e dos participantes. Na seqüência, ocorre o estudo de oito tópicos do curso (um por semana). Para cada tópico é feito um estudo individual, uma discussão assíncrona na Conferência e um bate-papo no Debate. Finalizando esta fase, os aprendizes são divididos em grupos para produzirem colaborativamente um conteúdo educacional. Os grupos são divididos com base na gestão de competências dos participantes. Por fim, os mediadores finalizam o curso e divulgam as notas finais.
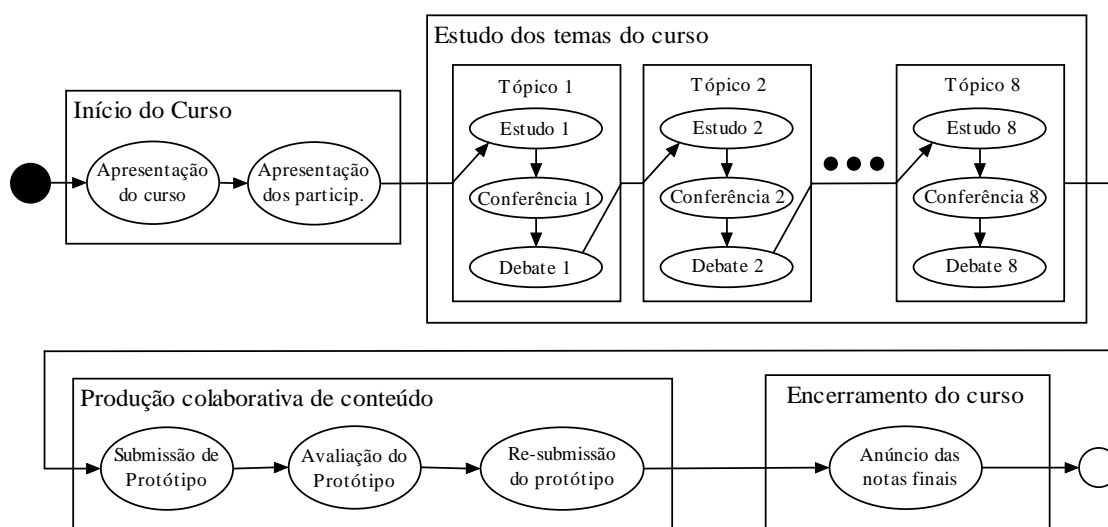
Figura 3.11. Seqüenciamento de atividades no curso TIAE

Cada uma das atividades representadas na Figura 3.11 é composta de tarefas. Gerenciar o fluxo entre as atividades e tarefas é parte da responsabilidade da coordenação. As tarefas que compõem as atividades possuem interdependências e necessitam de mecanismos de coordenação para acompanhar seu desenvolvimento. Diferentemente de fluxos de trabalho tradicionais, onde a não execução de uma tarefa pelo responsável causa a interrupção do fluxo, no curso TIAE há o fator do tempo que determina quando uma tarefa é declarada finalizada. Por exemplo, se um aprendiz não participar de uma determinada atividade durante uma semana, perde aquele tópico e, na semana seguinte, participa da discussão do próximo.

A Figura 3.12 apresenta as interdependências entre as tarefas de uma conferência. Há três papéis envolvidos nesta atividade: mediador, seminarista e aprendiz. O mediador seleciona o aprendiz que será o seminarista da semana e inicializa a sessão. O seminarista submete então o seminário e as três questões para discussão. Os aprendizes postam mensagens argumentando sobre as questões propostas. O mediador então finaliza a sessão e avalia as mensagens.
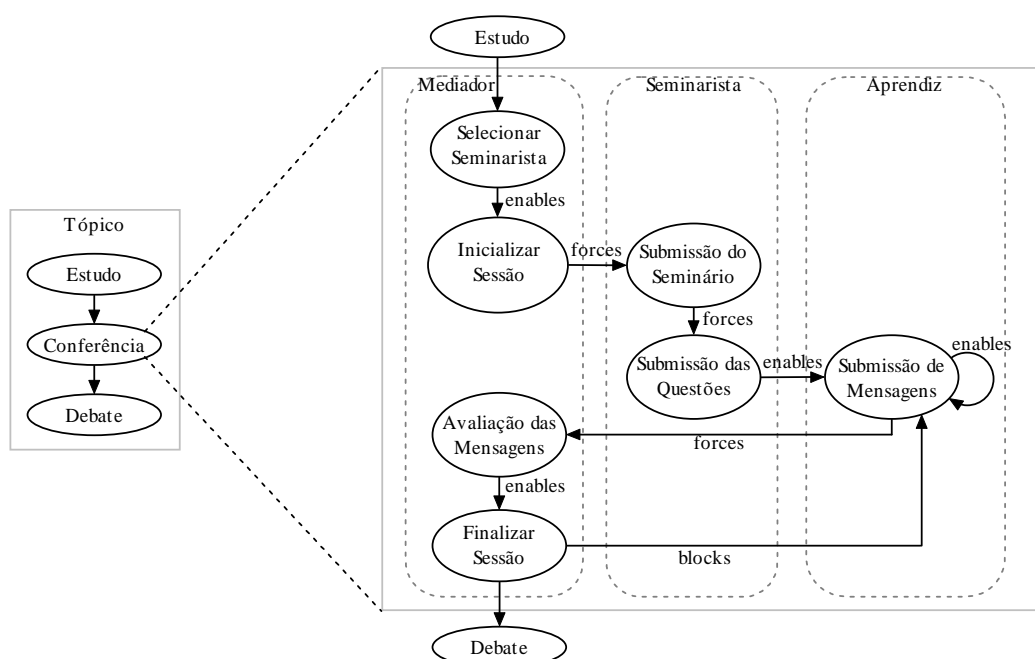
Figura 3.12. Interdependência entre as tarefas de uma conferência

As interdependências entre as tarefas apresentadas na Figura 3.12 são expressas em termos dos operadores *enables*, *forces* e *blocks*. A seleção do seminarista, por exemplo, habilita a inicialização da sessão, de modo que o mediador não inicia a sessão sem antes escolher o seminarista. Entretanto, a

escolha do seminarista não força a inicialização da sessão. A mesma relação acontece entre a submissão das questões pelo seminarista e a submissão de mensagens pelos aprendizes. O operador *blocks* é utilizado, por exemplo, para que após a finalização da sessão, o aprendiz não possa mais enviar mensagens. As interdependências entre as tarefas são caracterizadas pelo cronograma estabelecido e pelo adequado seqüenciamento de mensagens. Na versão atual do AulaNet, não há mecanismos de coordenação explícitos para a maior parte das tarefas, de modo que a coordenação fica a cargo do protocolo social, certificado pelos mediadores e pelo seminarista da semana, e embasado no acompanhamento da participação. Algumas interdependências são controladas por mecanismos de coordenação, como o de travar o envio de mensagens, que é utilizado pelo mediador ao inicializar e finalizar a sessão.

Durante o gerenciamento da execução de tarefas, principalmente daquelas que não foram precisamente definidas durante a pré-articulação, as informações de percepção são fundamentais. Este tipo de tarefa é comum em atividades ligadas à aprendizagem, onde os aprendizes tomam decisões e resolvem problemas sem o conhecimento completo do domínio [Simon, 1996]. Nestas tarefas a divisão e a organização do trabalho acontecem dinamicamente através da coordenação de atividades [Gross, 1997]. As informações de percepção transmitem as mudanças de planos, contribuindo para gerar um novo entendimento. Além disto, informam os participantes de aspectos temporais e espaciais de suas ações e facilitam a sincronização das tarefas individuais. As informações de percepção são apresentadas no espaço compartilhado e, muitas vezes, são provenientes do registro da informação. Na seção seguinte as informações de percepção do AulaNet são tratadas em mais detalhes.

A avaliação dos aprendizes no TIAE é feita através da participação e da qualidade das contribuições feitas em todo o curso. Cada mensagem do seminário é avaliada e comentada individualmente, objetivando orientar os aprendizes na construção do conhecimento e na formulação do texto, evitando que sejam enviadas contribuições que não agreguem valor ao grupo. Os problemas encontrados nas contribuições são comentados na própria mensagem, geralmente de forma visível a toda a turma. Os relatórios de participação embasam o acompanhamento da participação dos aprendizes nos diversos eventos do curso e

possibilitam a apreciação da qualidade da contribuição gerada por esta participação, do ponto de vista do docente [Fuks et al., 2003]. Estes relatórios dão indícios de quem não está participando e de quem está participando inadequadamente, tanto quantitativamente quanto qualitativamente, conforme ilustrado na Figura 3.13.

AULANet Tecnologias da Informação Aplicadas à Educação

Opções

Participantes	Lista de Discussão **0** (0)	Conferências **8.99** (6)	Debate **8.08** (2)	Tarefas **7.82** (2)	Co-autoria de Aprendiz **0** (0)	**Conceito Médio 8.57 (10)**
Alberto	Sem Conceito	Bom / 8.68	Ativo(a) / 6.67	Bom / 9	Sem Conceito	8.34
Alexandre	Sem Conceito	Bom / 9.69	Muito Ativo (a) / 8	Sem Conceito	Sem Conceito	7.41
Andre	Sem Conceito	Bom / 9.56	Muito Ativo (a) / 10	Péssimo / 0	Sem Conceito	7.74
Andréa	Sem Conceito	Bom / 8.12	Ativo(a) / 6	Bom / 9	Sem Conceito	7.88
Bernardo	Sem Conceito	Bom / 9.17	Muito Ativo (a) / 7.86	Regular / 7	Sem Conceito	8.47
Bruno	Sem Conceito	Bom / 9.73	Muito Ativo (a) / 10	Bom / 9	Sem Conceito	9.64

Controle Voltar Atualizar

Figura 3.13. Relatório do acompanhamento da participação

Em um curso como o TIAE, onde a maior parte das atividades é realizada assincronamente, a pressão para responder é reduzida, o que faz com que os aprendizes sejam tentados a não cumprir as atividades do curso em função de outras tarefas de sua vida particular [Graham et al., 1999]. Os mediadores do curso atuam constantemente exigindo contribuições dentro do período estipulado e intervindo para evitar a dispersão. No curso TIAE, o seminário dura 50 horas: de 12h de segunda-feira às 14h de quarta-feira. A Figura 3.14 apresenta a freqüência de mensagens enviada para cada hora do seminário nas edições de 2002.1 a 2003.2.
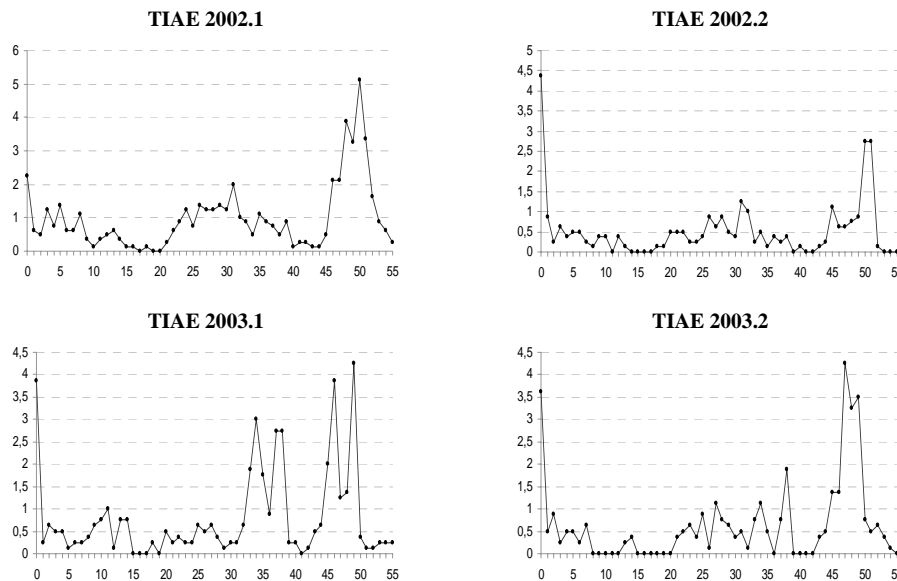
Figura 3.14. Freqüência média de mensagens para cada hora dos seminários das edições de 2002.1 a 2003.2

Pode-se notar na figura uma rajada durante as últimas 5 horas. Em alguns casos, mais de 50% das mensagens foram enviadas durante este período. A realização da tarefa no último momento possível é parte da “Síndrome do Estudante” [Goldratt, 1997]. Enviar contribuições próximo ao limite do prazo de encerramento do seminário dificulta o aprofundamento da discussão, já que estas mensagens dificilmente serão avaliadas e respondidas durante a discussão. Esta pode ser a razão para a quantidade excessiva de folhas nas árvores de alguns seminários e a conseqüente baixa interatividade. Para evitar este comportamento indesejado, os mediadores passaram a encorajar o envio mais cedo de contribuições. Entretanto, o simples encorajamento não funcionou. Na edição de 2004.1 foi adotada a seguinte regra: se até a 25ª hora de seminário o aprendiz não enviar metade da quantidade esperada de mensagens, as notas de suas mensagens subseqüentes passam a ser divididas pela metade. Esta regra foi estabelecida nos últimos 4 seminários. Conforme ilustrado na Figura 3.15, houve uma maior distribuição de mensagens e a rajada de mensagens ao final do período foi reduzida. O percentual de mensagens enviadas nas últimas 5 horas caiu de 33% na primeira metade do curso para 13% na segunda metade. Esta redução também foi observada nas edições subseqüentes do curso.
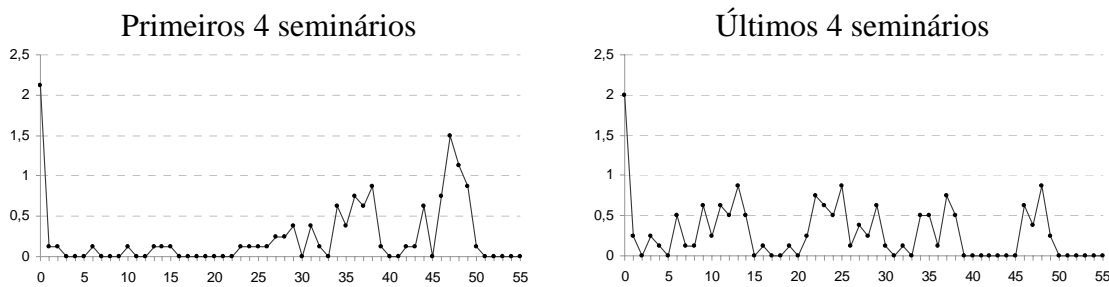
Figura 3.15. Freqüência média de mensagens para cada hora dos seminários para a edição 2004.1

Comunicação e coordenação, apesar de vitais, não são suficientes. É necessário espaço compartilhado para criar entendimento compartilhado [Schrage, 1995]. Os compromissos são assumidos durante a comunicação, e a coordenação gerencia as tarefas necessárias para cumprir os compromissos. Porém, para que as tarefas sejam realizadas em grupo é necessário espaço compartilhado.

3.6. Cooperação: Produção no Espaço Compartilhado

De acordo com o dicionário Houaiss [2001], cooperar é:

> atuar, juntamente com outros, para um mesmo fim; contribuir com trabalho, esforços, auxílio.

No trabalho em grupo, cooperação é a operação conjunta dos participantes no espaço compartilhado, visando a realização das tarefas. Durante a cooperação, os participantes produzem, manipulam, refinam e organizam objetos, como documentos, planilhas, gráficos, etc. Para atuar nos objetos, os participantes contam com mecanismos de expressão, e para se informar dos resultados de suas atuações (*feedback*) e das ações de seus colegas (*feedthrough*) dispõem de informações de percepção. Os participantes usam estas informações para planejar as interações subseqüentes [Neisser, 1976].

A cooperação é síncrona, como nos whiteboards, ou assíncronas, como nos repositórios de arquivos; fracamente acoplada, como na linha de montagem, ou fortemente acoplada, como nos editores cooperativos [Streitz et al., 1992]. Algumas ferramentas disponibilizam um espaço privativo para que o indivíduo trabalhe antes de se expor e oferecem suporte à transição entre o espaço privado e o coletivo. Eventualmente, algumas informações do espaço privado são

disponibilizadas como feedthrough, informando, por exemplo, que o indivíduo está digitando, a posição do indivíduo no espaço compartilhado, etc.

Os elementos da cooperação são relacionados ao registro e recuperação dos objetos e ações. O registro da informação visa aumentar o entendimento entre as pessoas, reduzindo a incerteza (relacionada com a ausência de informação) e a equivocalidade (relacionada com a ambigüidade e com a existência de informações conflitantes) [Daft & Lengel, 1986]. Os indivíduos trabalham as informações e se comunicam para solucionar os desentendimentos.

Preservar, catalogar, categorizar e estruturar os objetos produzidos pelos participantes é uma forma de estabelecer a memória do grupo [Borges et al., 2000]. O conhecimento informal, isto é, idéias, fatos, questões, pontos de vista, conversas, discussões e decisões decorrente da interação ao longo do processo, é difícil de ser capturado e registrado. Entretanto, este patrimônio interacional possibilita recuperar o histórico da discussão e o contexto no qual as decisões foram tomadas, bem como analisar a interação com o intuito de refinar a dinâmica da colaboração [Siebra et al., 2005]. Registrar o raciocínio que levou a um determinado artefato (*design rationale*) também possibilita averiguar, em um novo contexto, se os motivos pelos quais as decisões de projeto foram tomadas continuam válidos, embasando a tomada de decisões [Andrade et al., 2002].

Há ferramentas na literatura que utilizam o hipertexto para a organização da memória do grupo [Shum & Hammond, 1994]. Algumas possibilitam ligar os artefatos ao espaço compartilhado, explicitando nestas ligações as interações que os originaram. Os contextos dos artefatos e das interações são preservados, facilitando o seu entendimento e a posterior recuperação, servindo de base para uma etapa de pós-articulação. A memória do grupo passa a ser formada pelos artefatos (memória do produto) e pelas redes de informações compostas pelos fatos, hipóteses, restrições, decisões e argumentos (memória do processo). Por exemplo, o Evolving Artifact [Ostwald, 1995], voltado para o desenvolvimento de software, integra documentação baseada em hipertexto com protótipos e possibilita, a partir da documentação, executar o protótipo. Os usuários registram seus comentários e críticas ao interagir com o software. O artefato-protótipo possibilita que os envolvidos no processo do desenvolvimento reflitam sobre as conseqüências do projeto [Schön, 1983; Schön & Bennet, 1996].

Algumas ferramentas possibilitam que os participantes avaliem e ranqueiem os objetos presentes no repositório compartilhado, de modo a dar subsídio a um sistema de recomendação [Motta & Borges, 2000]. Algumas ferramentas, como o CVS, oferecem facilidades de gerenciamento de versão, possibilitando recuperar edições anteriores, comparar versões, atribuir comentário a cada versão, etc. Esta funcionalidade é especialmente útil quando a cooperação envolve vários participantes, onde é necessário identificar o autor, a data e o propósito de cada modificação realizada nos objetos. O registro propicia a recuperação e auditoria em caso de problemas.

A produção é dependente de como o espaço compartilhado está estruturado para apresentar os objetos de cooperação e a interação. Na interação entre pessoas e ambiente em uma situação face-a-face, a obtenção de informações é rica e natural, visto que os sentidos são usados em sua plenitude [Fitzpatrick et al., 2002]. Em ambientes digitais, os meios de transmitir as informações aos órgãos sensoriais dos seres humanos são mais restritos. Por outro lado, em um ambiente digital, os eventos são filtrados de modo a reduzir dispersões com informações irrelevantes, que normalmente permeiam uma colaboração face-a-face [David & Borges, 2001]. As informações de percepção estabelecem o contexto de trabalho e propiciam antecipar ações e necessidades, bem como identificar as intenções dos companheiros do grupo, para assisti-los quando for possível e necessário [Baker et al., 2001]. As informações de percepção também ajudam a identificar o papel e as tarefas de cada um com relação às metas da colaboração e com os objetos da cooperação [Gutwin et al., 1995]. Através da percepção, os indivíduos tomam ciência das mudanças ocorridas no ambiente, redirecionam suas ações e prevêem possíveis necessidades [Neisser, 1976].

O projetista de um ambiente virtual identifica quais informações de percepção são relevantes, como são obtidas, onde elementos de percepção são necessários, como exibi-los e como fornecer aos indivíduos controle sobre o fluxo de informações e sobre questões relativas à privacidade. Para evitar a sobrecarga, é necessário balancear a necessidade de fornecer informações com a de preservar a atenção sobre o trabalho, e fornecer informações na forma assíncrona, estruturada, filtrada, agrupada, resumida e personalizada [Kraut & Attewell, 1997]. Uma visão geral é fornecida para que o indivíduo selecione em que parte

da informação deseja trabalhar, e mais detalhes são obtidos quando forem demandados. O espaço compartilhado é projetado de modo que a percepção apóie o trabalho em grupo e o estabelecimento do contexto de trabalho [Borges et al., 2004].

Um projeto adequado dos elementos de percepção a serem utilizados em uma aplicação disponibiliza as informações que os participantes necessitam para prosseguir seu trabalho, de modo a reduzir as interrupções a colegas para solicitar informações já disponíveis no ambiente [Segal, 1994]. Entretanto, não é possível ao projetista definir a priori quais elementos de percepção serão adequados e suficientes. Este processo deve ser contínuo e experimental para que os elementos sejam adaptados às necessidades dos indivíduos. Como cada um tem suas capacidades, necessidades e preferências, os elementos devem ter flexibilidade o suficiente para se adequarem às diferentes personalidades.

3.6.1. Estudo de Caso da Cooperação no AulaNet e no Curso TIAE

O serviço Conferências provê um espaço compartilhado de informações onde os aprendizes cooperam produzindo e refinando conhecimento através de um processo argumentativo. Os aprendizes produzem objetos de cooperação, neste caso, mensagens da conferência. No espaço compartilhado da Conferência, são apresentadas informações de percepção sobre os objetos de cooperação, incluindo a autoria, a data, a categoria, o assunto e o conceito dado pelo mediador do curso. As interações do grupo são registradas, categorizadas e estruturadas nas mensagens, que representam os objetos de cooperação. A memória do grupo é preservada nas idéias, fatos, questões, pontos de vista, conversações, discussões e decisões, indicando o histórico da colaboração e o contexto onde a aprendizagem ocorreu [Siebra et al., 2005]. O registro também possibilita efetuar buscas no repositório de mensagens. Nas Conferências do AulaNet a busca é feita a partir das categorias, data, título, corpo, autor e indicação de leitura.

O controle remoto, apresentado na Figura 3.2, apresenta várias informações de percepção. Na parte superior encontra-se o código da disciplina, oferecendo um elemento de percepção individual de localização e contexto. Os itens do controle

remoto oferecem a percepção de quais são as opções disponíveis no momento para o participante. Ao lado de cada item do menu, há um botão circular que muda de cor para fornecer informações sobre os serviços. Um botão azul indica o serviço que o participante selecionou, indicando sua localização. Um botão laranja claro (em destaque na Figura 3.2) indica que existem possíveis ações a serem tomadas no serviço. Estas ações incluem a presença de um companheiro (nos serviços de comunicação síncronos) ou novos itens a serem trabalhados, como uma nova mensagem ou conteúdo. Ao passar o mouse sobre o botão aparece o total de itens sobre os quais é provável se tomar uma ação (itens não lidos, não resolvidos ou participantes conectados). Um botão laranja escuro indica um serviço sem novidades desde o último acesso. A partir destas informações de percepção, o aprendiz decide onde trabalhar. O controle remoto transfere para o aprendiz, até certo ponto, controle do processo de aprendizagem.

Algumas ferramentas oferecem suporte à análise estatística dos objetos e informações compartilhadas, de modo a possibilitar coletas e análises úteis para embasar a mediação do grupo. As informações sumarizadas são utilizadas para acompanhar a interação, reduzindo a necessidade de monitorar, ler e acompanhar as contribuições, que ocorrem em horários e freqüência variados [Gerosa et al., 2005]. Por exemplo, a Figura 3.16 apresenta a profundidade média, a porcentagem de folhas e a quantidade de mensagens em cada Conferência das edições 2002.1 e 2003.1 do curso TIAE. Na turma 2002.1 a profundidade média e a quantidade de mensagens diminuíram ao longo do tempo e o percentual de folhas aumentou, indicando menos interação na turma. Na edição 2003.1, ocorreu uma situação oposta: a profundidade média e a quantidade de mensagens aumentaram ao longo das conferências e a porcentagem de folhas diminuiu, indicando um aumento da interação. Estas informações foram obtidas a partir do registro das mensagens, que possibilita recuperar o contexto da cooperação no grupo.
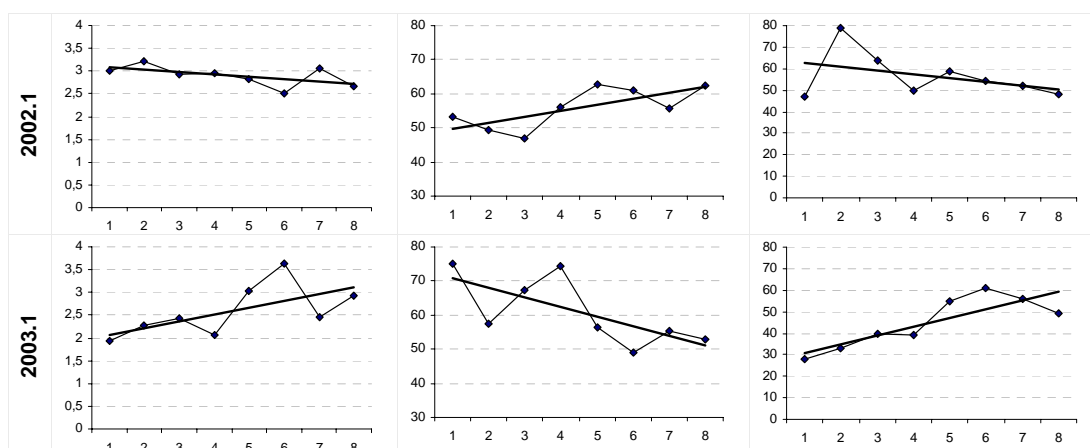
Figura 3.16. Profundidade média, porcentagem de folhas e quantidade de mensagens nas Conferências das edições de 2002.1 e 2003.1 do curso TIAE

O registro das mensagens também possibilita analisar o histórico em função do nível da árvore, do tamanho, da categoria e do horário de envio [Fuks et al., 2005]. A Figura 3.17 ilustra alguns gráficos utilizados para analisar as mensagens do curso TIAE. A figura apresenta a quantidade de mensagens por nível, o uso das categorias por nível, o tamanho da mensagem por categoria e a nota em função do tamanho. A partir destes gráficos o docente analisa o comportamento do grupo na cooperação.
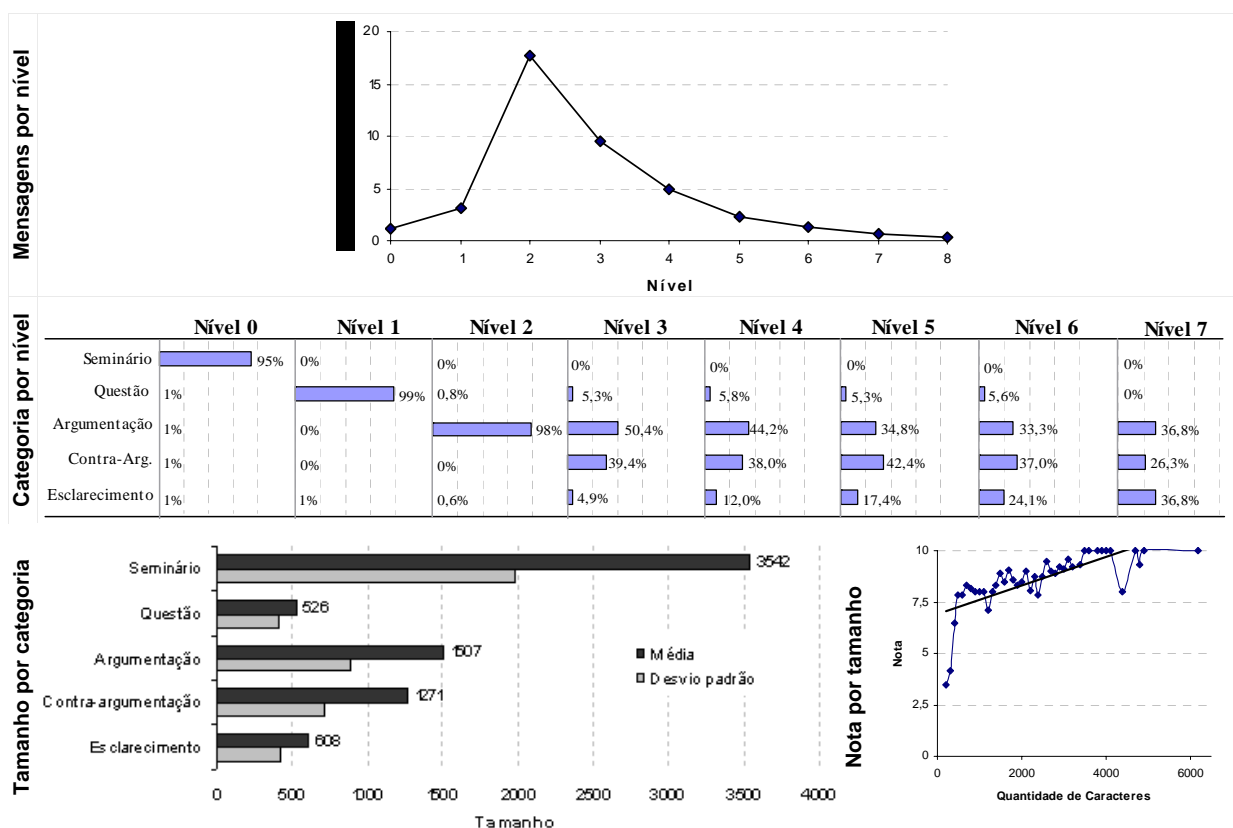
Figura 3.17. Profundidade média, porcentagem de folhas e quantidade de mensagens nas Conferências das edições de 2002.1 e 2003.1 do curso TIAE

Conforme esperado, no nível zero da árvore há em média 1 mensagem, que é da categoria Seminário. No nível um, 3 mensagens da categoria Questão. No nível dois, ocorre o pico da quantidade de mensagens, que são as respostas diretas às questões através da categoria Argumentação. No nível três em diante a quantidade de mensagens cai e aparecem as contra-argumentações. As mensagens da categoria Seminário são as maiores, enquanto as da categoria Questão são as menores, e mensagens com tamanho muito abaixo da média costumam receber uma avaliação ruim. Os gráficos da Figura 3.14 e da Figura 3.15 são exemplos de análises em função do horário de envio.

Este tipo de análise é especialmente útil para a versão PDA do ambiente AulaNetM. A apresentação de informações de caráter visual, perceptíveis em um relance, se mostrou adequada às telas reduzidas dos PDAs e às situações nas quais eles são comumente utilizados, como no caso de consultas rápidas no intervalo de uma atividade, na fila do restaurante ou no corredor à espera de uma reunião [Filippo et al., 2005]. A Figura 3.18 ilustra a exibição das mensagens de uma conferência na forma expandida, na forma de árvore e informações estatísticas sobre as características das mensagens.

Figura 3.18. Mensagens de uma conferência na forma expandida, na forma de árvore e informações estatísticas sobre as características das mensagens em um PDA

Na fase de produção colaborativa de conteúdos educacionais multimídia e interativos, a cooperação acontece na produção conjunta do documento. Para o desenvolvimento do conteúdo, os aprendizes utilizam suas ferramentas habituais e posteriormente submetem pelo ambiente os conteúdos produzidos. Os aprendizes são organizados em grupos de dois ou três, baseado no perfil que previamente

preencheram com seus interesses e qualificações em cada um dos tópicos do curso. Baseado neste perfil, o AulaNet sugere formações de grupo que melhor satisfaçam aos critérios definidos pelo mediador (grau de habilidade, interesse e performance) [Cunha et al., 2003]. Após a submissão do conteúdo, inicia-se uma fase de avaliação pelos próprios colegas. Membros de outros três grupos escolhidos avaliam o conteúdo submetido em conferências criadas para cada grupo, onde os aprendizes discutem os problemas encontrados nos protótipos. Após a conclusão do período de discussão, os grupos têm um prazo para submeter uma versão revisada, incorporando as contribuições dos colegas. Esta versão revisada é avaliada pelo coordenador do curso para eventualmente ser incorporada ao repositório.

A percepção interconecta os elementos do modelo 3C [Gerosa et al., 2003]. A partir da percepção, os participantes identificam as necessidades de trabalho [Greenberg, 2003]. No serviço Mensagem aos Participantes, é iniciada uma comunicação síncrona através de mensagens instantâneas a partir da indicação de quem está presente no ambiente. Quando um aprendiz submete um conteúdo educacional pelo serviço Co-autoria de Aprendiz o coordenador do curso é notificado para avaliar, trocar mensagens com o aprendiz e, eventualmente, incorporar o conteúdo ao repositório do curso.

3.7.
Classificação de Acordo com o Modelo 3C

O modelo 3C guia a especificação de uma nova ferramenta e a análise do suporte computacional de uma existente. Nesta seção, alguns exemplos são analisados com o intuito de clarificar alguns critérios de classificação e conceitos referentes ao modelo 3C de colaboração.

O serviço Conferências do AulaNet possibilita a produção de mensagens e seu posicionamento em árvores, registro, busca e compartilhamento. Mesmo tendo este ferramental voltado à cooperação, o propósito principal do serviço é a comunicação. Os participantes não utilizam o serviço com o intuito de construir árvores e mensagens, e sim construir e discutir conhecimento através da argumentação. O conhecimento negociado não é explicitamente representado no

serviço, de modo que o serviço é classificado como comunicação. Em uma dinâmica de trabalho, ele é utilizado para dar suporte computacional à conversação entre os participantes.

Uma ferramenta projetada para a comunicação pode ser utilizada para coordenação. Por exemplo, em alguns casos um coordenador utiliza uma videoconferência, que tipicamente é uma ferramenta de comunicação, para monitorar o que seus coordenados estão fazendo. Definir um propósito específico para a ferramenta possibilita construir um suporte computacional mais apropriado à atividade. Um repositório de arquivos, que tipicamente é um serviço de cooperação, por exemplo, ao ser utilizado como um guichê de recebimento de tarefas e municiado com funcionalidades específicas a este propósito é re-classificado como coordenação.

Serviço	Descrição	Classificação
Fóruns de Discussão	Tópicos que estão em discussão de maneira hierárquica	Comunicação
Bate-Papo	Conversa em tempo-real entre os alunos do curso e os formadores	Comunicação
Correio	Sistema de correio eletrônico interno ao ambiente	Comunicação
Dinâmica do Curso	Informações sobre a metodologia e a organização do curso	Coordenação
Agenda	Programação de um determinado período do curso (diária, semanal, etc.)	Coordenação
Avaliações	Avaliações em andamento no curso	Coordenação
Atividades	Atividades a serem realizadas durante o curso	Coordenação
Exercícios	Exercícios com questões dissertativas, de múltipla-escolha, de associar colunas e de verdadeiro ou falso	Coordenação
Grupos	Gerenciamento de subgrupos para o desenvolvimento de tarefas	Coordenação
Perfil	Espaço para que cada participante do curso se apresente aos demais	Coordenação
Acessos	Acompanhamento da freqüência de acesso dos participantes ao curso e às ferramentas	Coordenação
Intermap	Visualização da interação dos participantes do curso nas ferramentas de comunicação	Coordenação
Material de Apoio	Informações úteis relacionadas à temática do curso	Cooperação
Leituras	Artigos relacionados à temática do curso	Cooperação
Perguntas Freqüentes	Perguntas realizadas com maior freqüência durante o curso e suas respectivas respostas	Cooperação
Parada Obrigatória	Conteúdos que visam desencadear reflexões e discussões entre os participantes ao longo do curso	Cooperação
Mural	Espaço reservado para que todos os participantes disponibilizem informações relevantes	Cooperação
Diário de Bordo	Espaço para que cada participante registre suas experiências ao longo do curso, eventualmente de forma compartilhada	Cooperação
Portfólio	Armazenamento de textos, arquivos e endereços da Internet, de maneira privada ou pública	Cooperação

Tabela 3.3. Serviços do ambiente TelEduc

Para exemplificar a análise em função do modelo 3C, foi utilizado o ambiente TelEduc (http://teleduc.nied.unicamp.br). Visando a colaboração em um

ambiente de ensino-aprendizagem, são disponibilizados aos participantes diversos serviços. Os serviços, suas descrições e a classificação em função do modelo 3C é apresentada na Tabela 3.3.

O TelEduc apresenta os serviços Fórum de Discussão, Bate-Papo e Correio voltados para a comunicação. Estes serviços oferecem suporte à troca de mensagens e argumentação. Os serviços Dinâmica do Curso, Agenda e Atividades são serviços de coordenação, pois objetivam a pré-articulação das tarefas, realizada pelo professor do curso. Os serviços Avaliações, Exercícios, Acessos e Intermap são voltados para o professor acompanhar a participação quantitativa e qualitativa no ambiente, sendo classificados também como coordenação. O serviço Grupo possibilita organizar os participantes em subgrupos para a realização das atividades, sendo também de coordenação. Os serviços Material de Apoio, Leituras, Perguntas Freqüentes e Parada Obrigatória são serviços através dos quais o professor disponibiliza conteúdos didáticos para os aprendizes, sendo voltados para a cooperação. Os serviços Diário de Bordo, Mural e Portfólio são serviços onde os aprendizes disponibilizam conteúdos de maneira privada ou compartilhada, também sendo serviços de cooperação.

Figura 3.19. Serviço Fórum de Discussão do TelEduc

Ao analisar separadamente cada um destes serviços também são encontradas funcionalidades de comunicação, de coordenação e de cooperação. Por exemplo, no serviço Fórum de Discussão, cuja interface está apresentada na Figura 3.19, são encontradas funcionalidades de envio de mensagens (comunicação), controle

de permissões para escrita e leitura ou somente leitura (coordenação), integração com o serviço de avaliação (coordenação), busca (cooperação), diferentes ordenações para a lista de mensagens (cooperação), registro em arquivo (cooperação) e lixeira para fóruns removidos (cooperação).
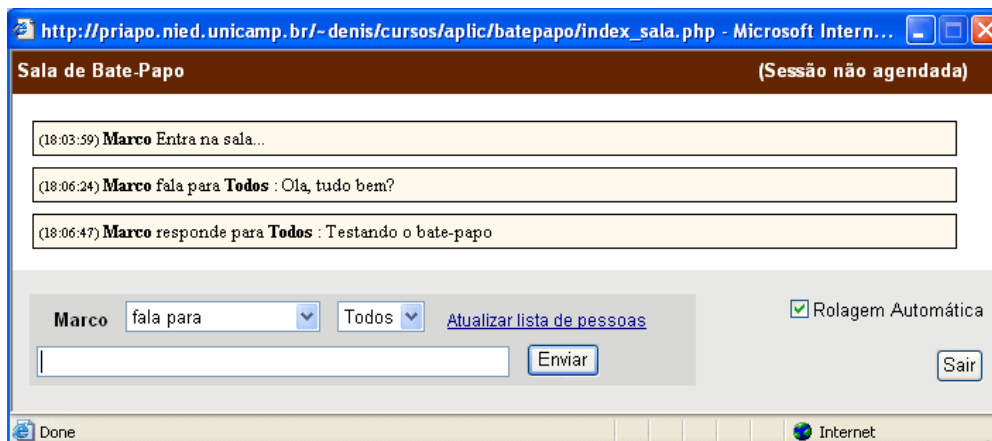
Figura 3.20. Serviço Bate-Papo do TelEduc

O serviço Bate-Papo, apresentado na Figura 3.20, apresenta funcionalidades de envio e de recebimento de mensagens (comunicação), categorização de mensagens (comunicação), gerenciamento de sessões (coordenação), lista de participantes (coordenação), registro (cooperação) e busca (cooperação).
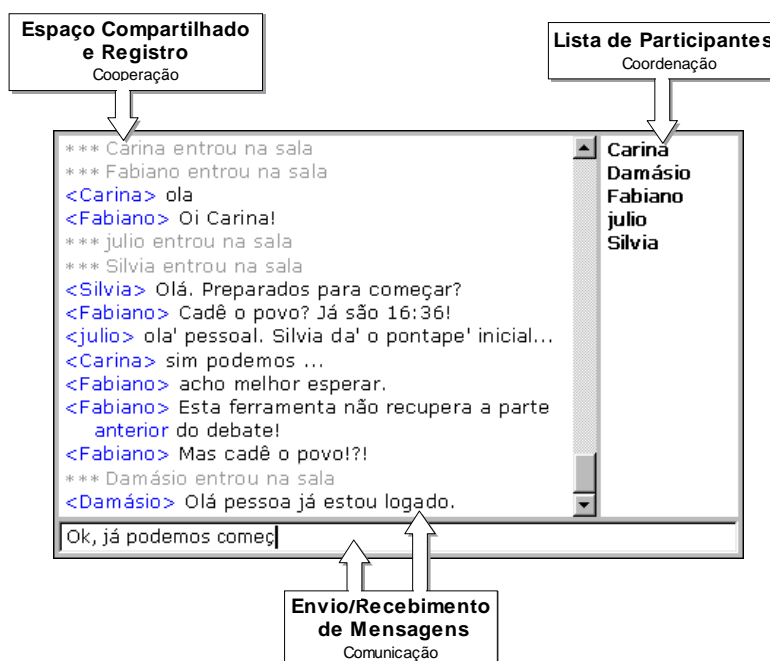
Figura 3.21. Serviço Bate-Papo do AulaNet

O serviço Bate-Papo do AulaNet, apresentado na Figura 3.21, apresenta uma área para envio e recebimento de mensagens (comunicação), uma lista de

participantes (coordenação) e um espaço compartilhado e registro de mensagens (cooperação).

3.8. Outros Modelos de Colaboração

Nesta seção são descritos sucintamente alguns outros modelos voltados para o trabalho colaborativo encontrados na literatura. Alguns modelos são voltados para um determinado tipo de trabalho em grupo e outros são mais aprofundados em um determinado elemento da colaboração.

A teoria das atividades [Kuutti, 1991] considera as atividades como sendo as unidades básicas de análise da colaboração. Neste modelo, uma atividade é executada através de ações que transformam objetos. Os participantes são organizados em comunidades e são classificados em ativos e passivos, dependendo de seu envolvimento com o propósito da atividade. Para executar as ações, os participantes utilizam ferramentas que agem sobre os objetos, e regras definem os relacionamentos entre os participantes nas comunidades. O enfoque dado à teoria das atividades é distinto do modelo 3C, que trata a modelagem de tarefas sob o enfoque da coordenação. van der Veer et al. [1996], Fitzpatrick et al. [1995], Kreifelts et al. [1993], Teege [1996] e Farias [2004] também modelam a colaboração a partir das tarefas.

Tripathi et al. [2002] propõem um middleware baseado em um modelo de colaboração, voltado para a gestão de segurança em sistemas colaborativos. O modelo adotado, chamado de *Role Based Collaboration Model*, enfoca o aspecto da coordenação do trabalho em grupo. Neste modelo, os participantes são representados por papéis e os papéis são associados às tarefas. As tarefas são vistas como operações, que incluem manipulação de objetos compartilhados e sincronização de ações. As operações possuem pré-condições, que são gerenciadas pelo middleware. Para descrever os papéis são definidas *role admission constraints*, que especificam as condições necessárias para um participante assumir um papel, *activation constraints*, que definem em quais papéis o participante deve ou não deve estar para assumir um papel, e *role validation*, que define as pré-condições comuns a todas operações associadas ao

papel. As pré-condições são definidas utilizando um modelo de especificação de eventos. As tarefas e os papéis são representados no contexto de uma atividade, que representa uma sessão de colaboração. Uma atividade é estruturada hierarquicamente, de modo que múltiplas atividades são aninhadas.
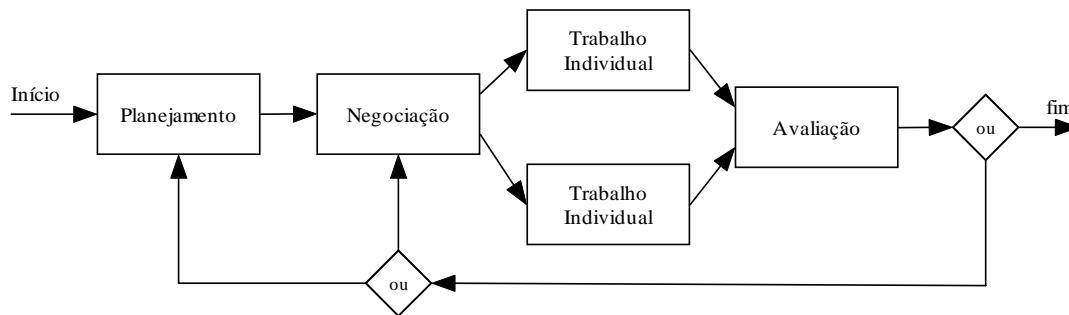
Figura 3.22. Modelo de colaboração proposto por Liu et al. [2001]

Liu et al. [2001] propõe um modelo de colaboração no contexto de um projeto colaborativo, ilustrado na Figura 3.22. Os autores abordam o aspecto temporal da colaboração, separando-a em atividades seqüenciais. A colaboração é vista como planejamento, negociação, trabalho individual e avaliação. Neste modelo, a interação é prevista apenas nas atividades de planejamento, avaliação e negociação, que são tratadas na coordenação (pré e pós-articulação) e comunicação do modelo 3C. A cooperação vista como ação conjunta em objetos compartilhados não é tratada pelo modelo de Liu et al.

Simon [1997] propõe um modelo de colaboração, chamado de DisNet, voltado para a construção de conhecimento a partir da comunicação. O modelo é dividido em um modelo de comunicação e um modelo de representação de conhecimento. A gestão e a distribuição de tarefas são tratadas no modelo de comunicação. Lange & Gershman [1992] propõem um modelo de colaboração para ser utilizado em um sistema de gestão do conhecimento. O modelo de colaboração proposto é baseado em três pilares: um modelo de tarefas, um modelo de arquivamento e recuperação de conhecimento e em um modelo de interação entre os participantes. O modelo apresenta alguma similaridade com o modelo 3C, pois a gestão de tarefas é contemplada na coordenação, a gestão de conhecimento, na cooperação, e a interação, na comunicação, entretanto, o modelo 3C possui outro enfoque e outros aspectos contemplados em seus elementos.

Alguns modelos de colaboração são voltados para ambientes educacionais, como por exemplo, o modelo utilizado na plataforma CLARE [Wan & Johnson, 1994], que organiza a colaboração nas atividades de sumarização, avaliação, comparação, argumentação e integração. Becker & Zanella [1998] propõem um modelo que define um processo, os papéis e os objetos manipulados em um ambiente de ensino-aprendizagem, considerando principalmente as atividades de discussão voltadas para a resolução de exercícios e compartilhamento de conhecimento.

Santoro et al. [2001] propõem um modelo de colaboração para embasar a construção de ambientes educacionais mais propícios para a aprendizagem colaborativa. O modelo, cujo diagrama encontra-se na Figura 3.23, é voltado para a aprendizagem baseada em projetos. Cada elemento do modelo é descrito por um sistema de padrões. A partir destes padrões, os requisitos de um ambiente de aprendizagem colaborativa são elicitados. O modelo é fundamentado no objetivo e no processo cooperativo que embasam o ambiente. O objetivo estabelece o contexto da aprendizagem e é relacionado ao conhecimento anterior, à teoria da aprendizagem, a aspectos culturais e às formas de cooperação. O processo cooperativo estabelece o estímulo para a aprendizagem e está relacionado às atividades, aos papéis, à memória (registro), à coordenação, à avaliação e à percepção. Cada um destes elementos é expandido em um modelo a parte. Os elementos dos modelos são tratados por padrões conceituais e, eventualmente, por padrões de projeto.
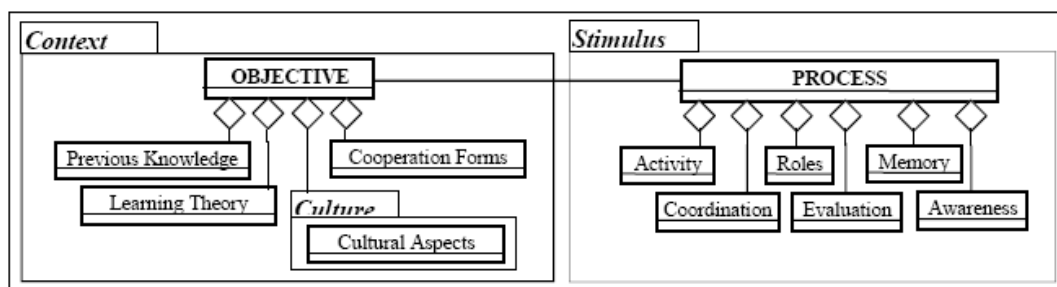
Figura 3.23. Modelo de colaboração proposto por Santoro et al. [2001]

Ellis & Wainer [1994] propõem um modelo para ser utilizado para classificação e comparação de groupware do ponto de vista de seus usuários. O modelo é baseado em três aspectos complementares: uma descrição dos objetos e das operações que são disponibilizadas para os usuários, uma descrição dos aspectos dinâmicos do sistema, como os fluxos de controle e de dados, e uma

descrição da interface com o usuário, tanto do ponto de vista do feedback quanto do *feedthrough*. São propostos modelos para cada um destes aspectos: *ontological model, coordination model* e *user interface model*, respectivamente. O *ontological model* possui alguns pontos em comum com o entendimento de cooperação do modelo 3C. Entretanto, alguns aspectos, como o controle de permissões e o controle de acesso, são tratadas na coordenação, do ponto de vista do modelo 3C. O *coordination model* lida com atividades, papéis, atores, objetivos, tarefas, procedimentos, etc., que também são tratadas pela coordenação no modelo 3C. A interface com o usuário não é tratada separadamente no modelo 3C. A percepção, que no modelo 3C é distribuída nos três Cs, no modelo de Ellis & Wainer é tratada no *user interface model*.

3.9. Considerações Finais

Colaborando, as capacidades, os conhecimentos e os esforços individuais se complementam. Trabalhar em grupo também traz motivação para o membro, pois seu trabalho estará sendo observado, comentado e avaliado por pessoas de uma comunidade da qual faz parte (seu grupo de trabalho) [Benbunan-Fich & Hiltz, 1999]. Ao expressar as idéias para se comunicar com os outros membros, o indivíduo trabalha ativamente seus conceitos, refletindo sobre os mesmos e refinando-os, ocasionando uma melhoria na qualidade do trabalho e do aprendizado [Schön, 1983].

O modelo 3C é utilizado freqüentemente na literatura como um meio de classificar sistemas colaborativos, como por exemplo em Borghoff & Schlichter [2000]. Nesta tese, o modelo 3C é utilizado para a modelagem do domínio, de modo a embasar o desenvolvimento de groupware. A experiência acumulada nos 8 anos de desenvolvimento e utilização do AulaNet, a pesquisa na literatura e a análise de ferramentas de colaboração possibilitaram refinar o modelo e direcioná-lo para o desenvolvimento de groupware. Na versão 2.0 do AulaNet, o modelo 3C é utilizado na organização dos serviços de colaboração e não impacta o desenvolvimento do sistema. Na versão 3.0, são utilizados componentes concebidos com base no modelo 3C de modo a estreitar a relação entre os conceitos do domínio e a organização da implementação. De acordo com Brna

[1998] o suporte computacional à colaboração é melhorado se houver um aumento de consciência do modelo de colaboração que está sendo utilizado.

O modelo 3C, como todo modelo, é uma simplificação da realidade e, portanto, não deve ser usado indistintamente em todas as situações. O objetivo do modelo não é representar a colaboração por si só, e sim a colaboração do ponto de vista do suporte computacional. O modelo foi concebido para ser usado como um guia para analisar o problema e organizar o desenvolvimento. O modelo foi refinado com o objetivo de embasar o desenvolvimento e construir uma arquitetura, um kit de componentes e um processo de desenvolvimento para instrumentar o desenvolvedor de groupware. A própria abordagem, que propõe a utilização de componentes de software para mapear a modelagem do domínio, propicia a evolução da solução de modo a acompanhar o refinamento do modelo de colaboração.

4
Montagem de Groupware e de Serviços Colaborativos

Neste capítulo é descrita a abordagem utilizada, discutindo os tipos de componentes adotados, os *component kits* propostos, a arquitetura e o modelo de componentes utilizados, a maneira de descrever os componentes, de instanciar groupware e de utilizar frameworks do domínio para complementar a solução proposta.

4.1. Componentes de Groupware e de Colaboração

Um groupware normalmente oferece ao participante um ambiente com um conjunto de ferramentas colaborativas, utilizadas nos diferentes momentos da colaboração. A Tabela 4.1 apresenta as ferramentas encontradas nos sistemas de groupware: AulaNet[8], TelEduc[9], AVA[10], WebCT[11] e Moodle[12], do domínio educacional, e GroupSystems[13], YahooGroups[14], OpenGroupware[15] e BSCW[16], voltados para o trabalho em grupo.

---

[8] http://www.eduweb.com.br
[9] http://teleduc.nied.unicamp.br
[10] http://ava.unisinos.br
[11] http://www.webct.com
[12] http://www.moodle.org
[13] http://www.groupsystems.com
[14] http://groups.yahoo.com
[15] http://www.opengroupware.org
[16] http://bscw.fit.fraunhofer.de
Data de consulta: 15 de janeiro de 2006

	Serviços de Comunicação							Serviços de Coordenação									Serviços de Cooperação												
	Correio	Lista de Discussão	Fórum	Mural	Brainstorming	Chat	Mensageiro	Agenda	Relat de Atividades	Acomp. da Particip.	Questionário	Tarefas	SubGrupos	Gerenc. de recursos	Orientação	Votação	Reposit. de Conteúdos	Quadro Branco	Busca	Glossário	Links	Jornal Cooperativo	Classificador	Wiki	Gerenc. de contatos	Revisão em pares	FAQ	Anotações	RSS
AulaNet	X	X	X			X	X		X	X	X	X	X			X	X				X							X	
TelEduc	X		X	X		X		X	X	X	X	X	X				X				X						X	X	
AVA	X	X	X	X		X		X	X	X	X	X			X		X			X	X						X	X	
WebCT	X		X			X		X	X		X	X				X	X	X	X	X	X							X	
Moodle			X			X	X	X	X	X	X	X	X			X	X		X	X	X	X		X		X	X		X
GroupSystems			X		X			X		X	X	X	X			X							X					X	
YahooGroups		X				X		X	X		X					X	X		X	X	X								X
OpenGroupware	X			X				X				X		X											X				
BSCW			X					X	X			X	X			X	X		X		X				X				

Tabela 4.1. Ferramentas colaborativas encontradas em groupware

Conforme observado na tabela, diversas ferramentas similares são utilizadas em groupware. Por exemplo, a maioria dos sistemas analisados oferece fórum, chat, agenda, relatórios de atividades, questionários, gerenciamento de tarefas, votação, repositório e links. Cada ferramenta pode ser vista de forma relativamente isolada dentro do ambiente. Estas características são propícias à aplicação de técnicas de desenvolvimento baseado em componentes, onde as ferramentas colaborativas são os componentes do groupware. Os ambientes oferecem um conjunto de componentes que é instanciado e customizado para cada grupo e dinâmica da colaboração.

Nesta tese, as ferramentas colaborativas são chamadas de serviços. Conforme discutido na seção anterior, é possível classificar os serviços de acordo com seus propósitos e características em função do modelo 3C. Os serviços da Tabela 4.1 são organizados em serviços de comunicação, coordenação e cooperação. Em um ambiente baseado em componentes, a partir das necessidades da colaboração no grupo, o desenvolvedor seleciona os serviços mais adequados. Um modelo de componentes unificado para os diversos groupwares possibilita ao desenvolvedor selecionar dentre os diversos serviços de mesmo propósito o mais adequado. Conforme pode-se notar na Tabela 4.1, há serviços que são oferecidos em apenas um groupware, mas que, a princípio, poderiam ser disponibilizados para utilização nos demais. Para os usuários, é interessante intercambiar serviços, de modo a complementar os ambientes e reusar experiências.

Além dos groupwares possuírem serviços similares, os serviços possuem funcionalidades similares. Nos serviços dos diversos groupwares analisados, as funcionalidades são recorrentes. A Figura 4.1 apresenta os chats do Moodle e do WebCT. Pode-se notar que ambos possuem uma área compartilhada onde as mensagens são apresentadas, uma lista de participantes conectados e uma área para elaboração das mensagens. Ao utilizar a componentização, estas características são reusadas.
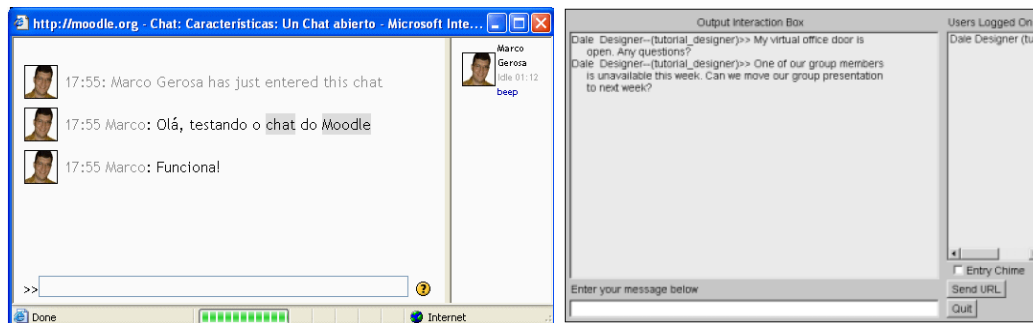
Figura 4.1. Chat do Moodle e do WebCT

Diferentemente dos demais, o chat do Moodle apresenta o tempo em que o participante está inativo, um recurso para chamar a atenção de um participante (beep) e a foto de cada um. O WebCT apresenta suporte para mensagens privadas e notificação de quando alguém entra na sala de bate papo. Encapsular estas funcionalidades em componentes possibilita ao desenvolvedor da aplicação disponibilizar para reuso os diversos aspectos do suporte à colaboração, de modo que outros desenvolvedores utilizem-nos na seleção das funcionalidades mais apropriadas para os grupos e atividades em questão.

O serviço Conferências e o serviço Correio para Turma do AulaNet, exibidos na Figura 4.2, compartilham o suporte ao envio, ao recebimento e à exibição de mensagens, à categorização, à avaliação da participação e ao bloqueio do canal de comunicação, entre outras funcionalidades. Encapsular as funcionalidades recorrentes em componentes propicia também o reuso do suporte computacional à colaboração de cada serviço, aumentando o reuso de código. Passa também a ser possível evoluir, ajustar e construir serviços variando e reconfigurando os componentes de colaboração.
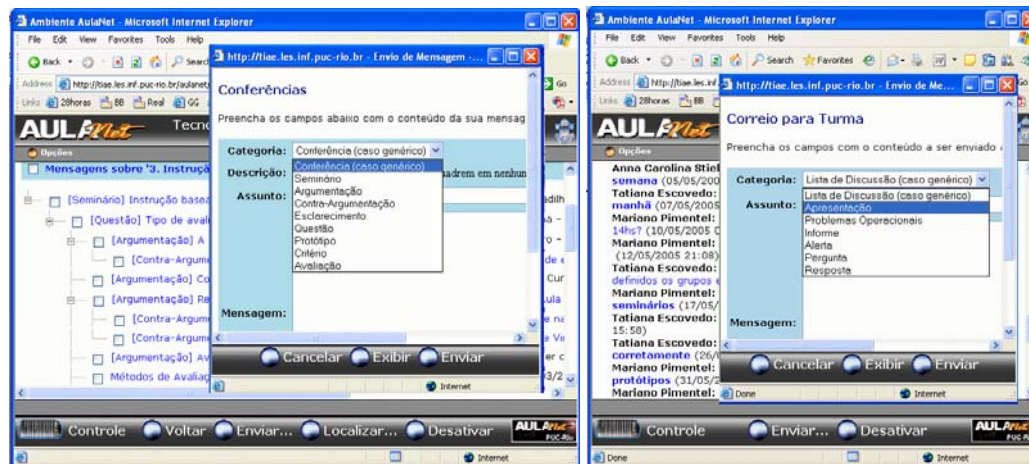
Figura 4.2. Serviços Conferências e Correio para Turma do AulaNet

A Figura 4.3 apresenta os serviços Bate Papo e Debate do AulaNet. A principal diferença entre estes dois serviços é o suporte à coordenação, que no primeiro é implementado apenas pela lista de participantes, enquanto no segundo é implementado pelo controle do espaço compartilhado, pelas técnicas de conversação e pela definição da ordem de participação. Em um serviço baseado em componentes, o suporte computacional à comunicação é compartilhado e os componentes que oferecem suporte computacional à coordenação são acrescentados, provendo reuso e especialização.
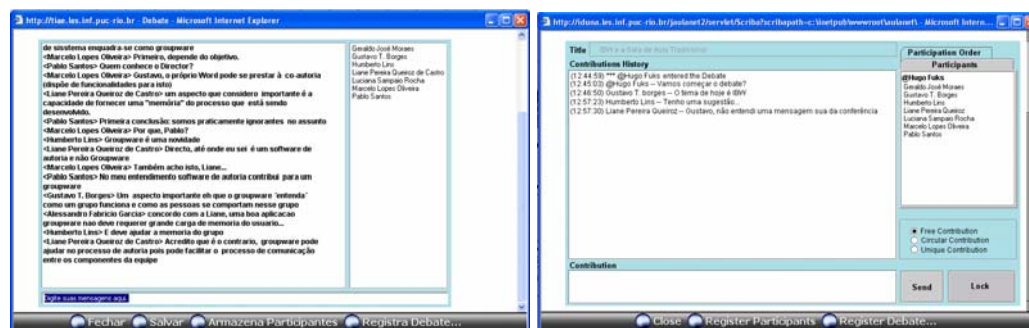
Figura 4.3. Serviços Bate-Papo e Debate do ambiente AulaNet

Estes cenários indicam a necessidade de se adotar a componentização de software em dois níveis. O primeiro nível contempla os componentes que provêem os serviços colaborativos, utilizados para oferecer suporte computacional à dinâmica da colaboração como um todo. O segundo nível contempla os componentes utilizados para montar os serviços, oferecendo suporte a determinados aspectos da colaboração dentro da dinâmica de uma atividade em particular. Na abordagem proposta, os componentes que implementam as ferramentas colaborativas são chamados de *serviços* e os componentes utilizados

para implementar o suporte computacional à colaboração dos serviços são chamados de *componentes de colaboração*.

Conforme ilustrado na Figura 4.4, um groupware é composto de serviços, que são reusáveis em diversos outros groupwares. Os serviços compartilham componentes de colaboração que implementam o suporte à colaboração, que nesta tese é modelada através do modelo 3C. A partir de component kits organizados em função do modelo 3C, o desenvolvedor monta uma aplicação para dar suporte à dinâmica da colaboração. Além do reuso, esta abordagem favorece também a capacidade de extensão da solução, ao possibilitar a inclusão de novos componentes.
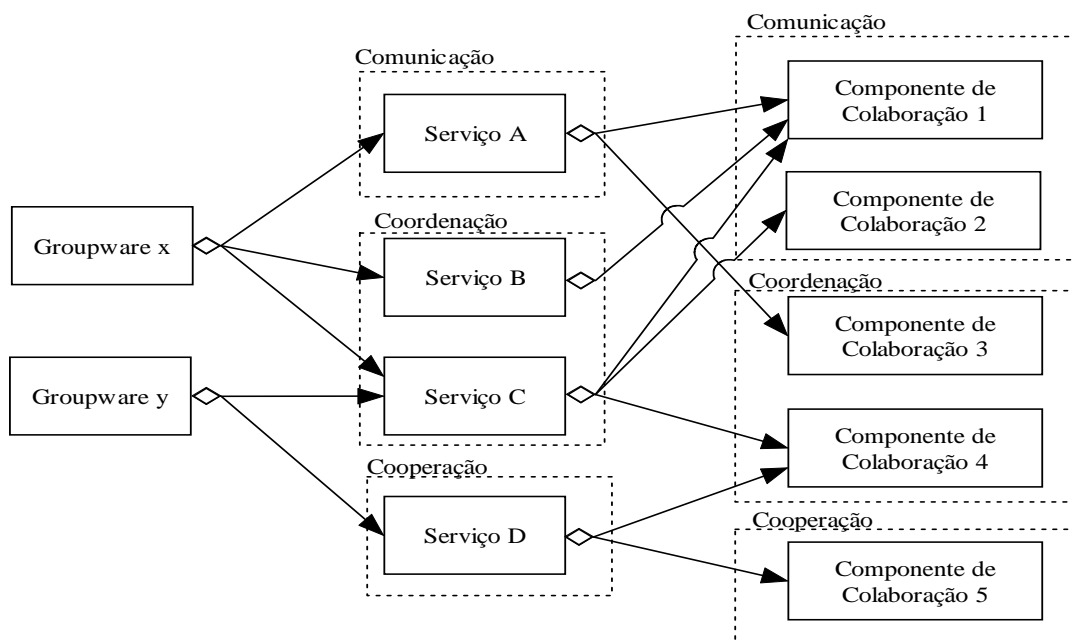
Figura 4.4. Groupwares montados a partir de serviços, e serviços montados a partir de componentes de colaboração

Conforme discutido no capítulo anterior, mesmo um serviço de comunicação, como um fórum de discussão, além dos componentes de comunicação, utiliza componentes de coordenação e de cooperação. Os componentes de colaboração de um C são reusados nos serviços dos demais Cs.

4.2.
O Collaboration Component Kit

O objetivo da abordagem proposta é instrumentar o desenvolvedor com *component kits* para serem usados na montagem dos groupwares e dos serviços colaborativos. Para obter o conjunto de componentes é necessária uma engenharia do domínio. A análise do domínio, primeiro passo da engenharia, foi elaborada neste trabalho com base na experiência do grupo desenvolvedor do AulaNet, que atua há 8 anos no desenvolvimento de ferramentas para colaboração, na literatura e na análise de ferramentas colaborativas. Entre outras, foram analisadas as ferramentas de comunicação dos diversos groupware apresentados na Tabela 4.1. A análise do domínio foi restrita às ferramentas de comunicação, que além dos elementos de comunicação, apresentam uma representatividade de elementos de coordenação e de cooperação.

A análise das ferramentas foi guiada pelo modelo 3C, sendo analisadas sob a perspectiva da comunicação, coordenação e cooperação. Os modelos de características foram diagramados na notação Odyssey-FEX, que é implementada no ambiente de modelagem Odyssey [Oliveira et al., 2005]. O Odyssey é um ambiente voltado para o suporte à engenharia de domínio e à engenharia de aplicações e oferece suporte computacional ao processo CBD-Arch-DE [Blois et al., 2004]. Na notação Odyssey-FEX um retângulo fechado simboliza uma característica obrigatória e um tracejado, uma característica opcional.

O modelo de características (*feature model*) apresentado na Figura 4.5 sumariza a análise do domínio da comunicação. Uma ferramenta de comunicação possui mensagens, que utilizam mídia textual, vídeo, áudio ou pictórica [Daft & Lengel, 1986]. Eventualmente, as mídias apresentam alguma variabilidade, como restrições ao tamanho do texto ou ao vocabulário disponível, no caso da mídia textual, e à taxa de captura e de envio de dados, no caso do áudio e vídeo. Algumas ferramentas disparam e-mail aos destinatários, possibilitam anexar arquivos às mensagens e disponibilizam um corretor ortográfico para auxiliar na elaboração do texto. Algumas ferramentas, como as Conferências e o Correio para Turma do AulaNet, oferecem a categorização de mensagens [Gerosa et al., 2001]. Também é encontrado em ferramentas de comunicação o suporte a carteiras de compromissos, como no ACCORD [Laufer & Fuks, 1995], e a caminhos de

conversação, como no Coordinator [Winograd & Flores, 1987]. As mensagens de uma ferramenta são organizadas em uma estrutura de diálogo, linear, hierárquica ou em rede [Stahl, 2001]. As mensagens são transmitidas em blocos ou continuamente.
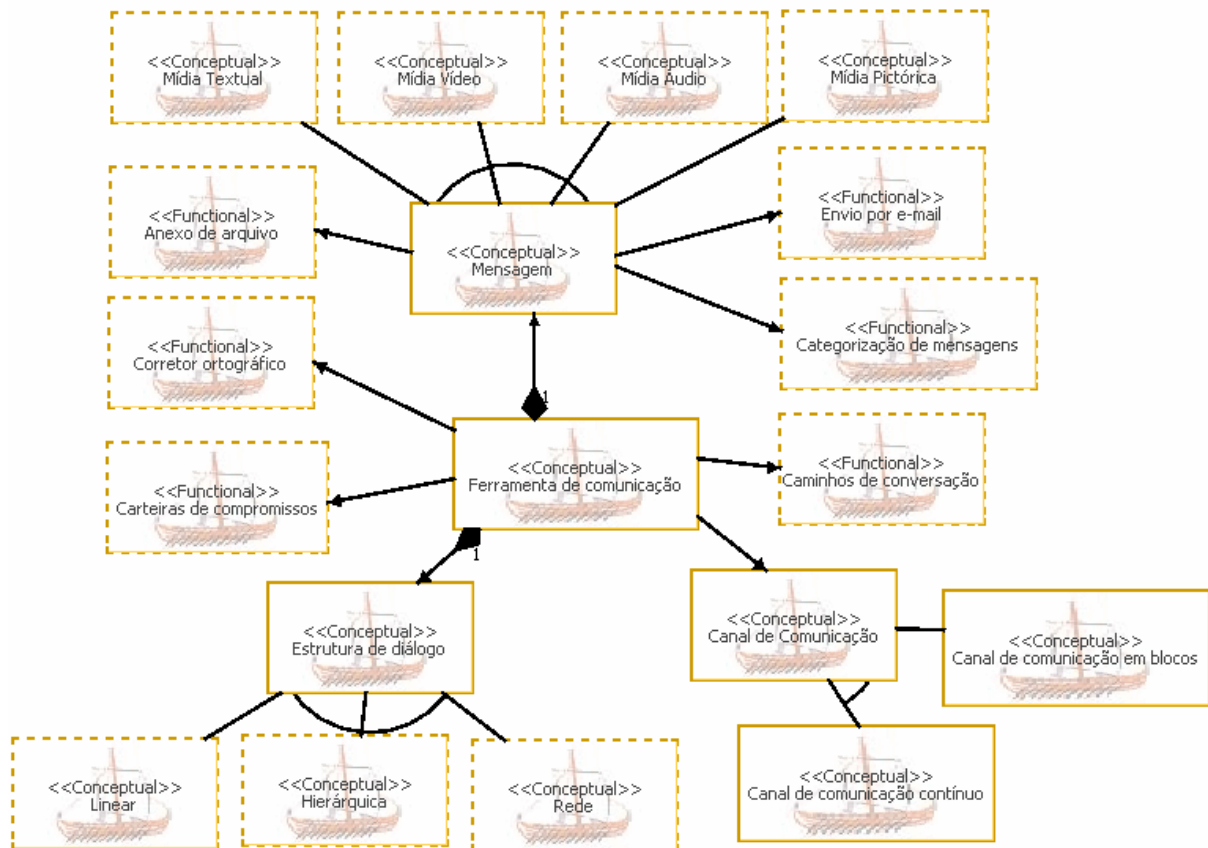
Figura 4.5. Modelo de características da comunicação nas ferramentas de comunicação

Conforme discutido no capítulo anterior, o suporte à coordenação em uma ferramenta de comunicação está relacionado às políticas de acesso ao canal, à gestão de tarefas e participantes e ao acompanhamento da participação. O modelo da Figura 4.6 sumariza a análise da coordenação nas ferramentas de comunicação. As ferramentas de comunicação apresentam um gerenciamento de permissões de acesso, associado a participantes ou a papéis. Os papéis são utilizados associados a tarefas, no escopo de uma atividade. Eventualmente, as tarefas são acompanhadas através de uma máquina de workflow. As participações dos indivíduos, principalmente em ferramentas de comunicação ligadas ao ensino-aprendizagem, são avaliadas, de modo a prover dados para o acompanhamento qualitativo da participação [Fuks et al., 2003]. A avaliação de mensagens provê subsídio para a gestão de competência dos participantes, que é utilizada para definir a dinâmica das atividades, a associação de papéis e a definição de

subgrupos. A participação ocorre no contexto de uma sessão e é embasada em informações de percepção, como por exemplo, a informação de que um determinado participante está escrevendo uma mensagem. Algumas ferramentas disponibilizam informações sobre a presença e a disponibilidade dos participantes, de modo a propiciar uma melhor sincronização da participação dos interlocutores. Um mecanismo de coordenação implementado por software encontrado em algumas ferramentas de comunicação, como o Debate do AulaNet, é o controle de palco, que possibilita implantar técnicas de conversação e políticas de acesso ao canal [Rezende et al., 2003].
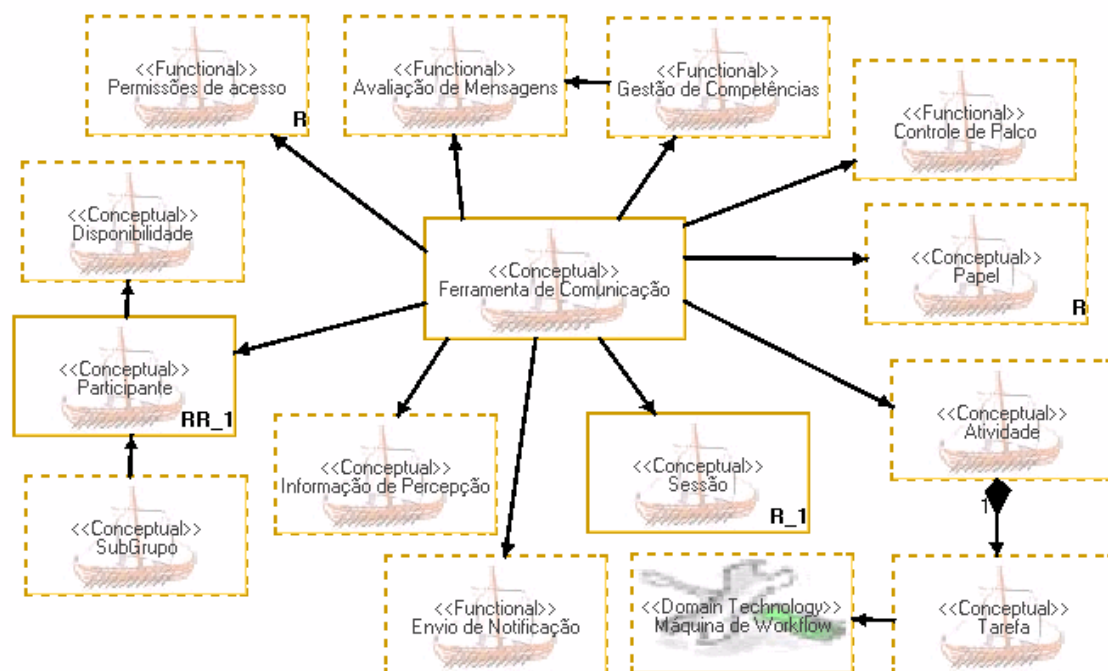
Figura 4.6. Modelo de características da coordenação nas ferramentas de comunicação

O suporte à cooperação em uma ferramenta de comunicação está relacionado ao registro e à manipulação das informações. O modelo de características da cooperação está apresentado na Figura 4.7. As mensagens ou as sessões das ferramentas de comunicação são registradas em repositórios na forma de objetos de cooperação. Estes objetos são associados a um gerenciamento de versões, a um registro de acesso, a análises estatísticas, a uma lixeira, a um sistema de recomendação, a um mecanismo de busca ou a um mecanismo de ranqueamento. As operações sobre os objetos são registradas na forma de um log, possibilitando a realização de auditoria e a volta a estados anteriores.
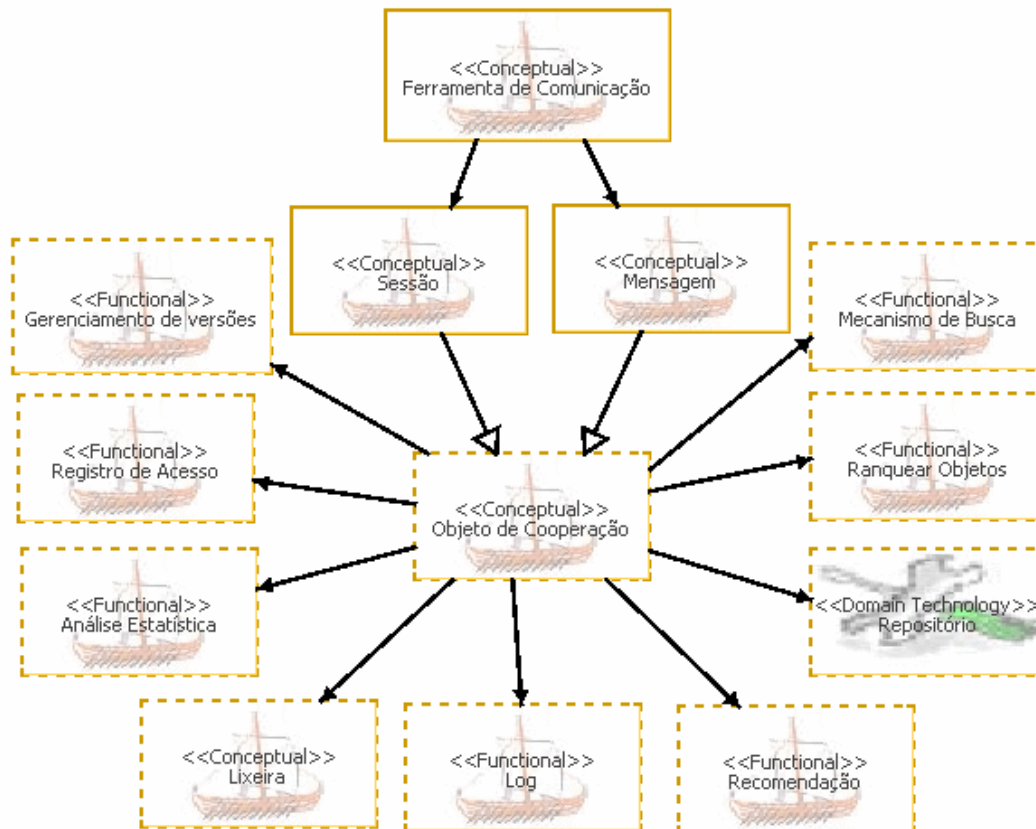
Figura 4.7. Modelo de características da cooperação nas ferramentas de comunicação

Os modelos de características são utilizados ao projetar uma nova aplicação para o domínio. As características são mapeadas em componentes que implementam as funcionalidades definidas. Os componentes são selecionados, implantados e configurados para oferecer suporte computacional às necessidades estabelecidas. Os componentes interoperáveis são organizados em *component kits*.

Conforme discutido no Capítulo 2, um *component kit* é uma coleção de componentes que foram projetados para trabalhar em conjunto [D'Souza & Wills, 1998]. De um kit de componentes gera-se uma família de aplicações, fazendo diferentes arranjos e eventualmente desenvolvendo alguns sob demanda. Nesta tese, para instrumentar o desenvolvedor de groupware, é oferecido o *Collaboration Component Kit*, com os componentes de colaboração utilizados para compor os serviços, implementando os aspectos da colaboração. Os componentes de colaboração foram obtidos a partir da análise do domínio, representada pelos modelos de características.

Conforme discutido no Apêndice A, um *component kit* não necessita ser exaustivo. Os *component kits* são extensíveis para acomodar novos componentes

necessários. No capítulo seguinte, algumas instanciações são analisadas. Componentes de software que sejam realmente reusáveis são refinados iterativamente até atingir a maturidade, a confiabilidade e a adaptabilidade desejadas [Gimenes & Huzita, 2005]. Os componentes de comunicação do *Collaboration Component Kit* são apresentados na Tabela 4.2.

Convencionou-se nomear os elementos arquiteturais na língua inglesa, pois no re-desenvolvimento do ambiente AulaNet, que serve de motivação para esta tese, o código está seguindo a padronização adotada pelos sistemas de código aberto, que são escritos em inglês, de modo a prepará-lo para futuras colaborações internacionais. O Apêndice B apresenta a descrição completa de todos os componentes propostos no kit. Para cada componente são descritos: nome, intenção, aplicabilidade, variabilidade, estrutura, usos conhecidos e componentes relacionados. Os componentes foram nomeados seguindo a recomendação de Cheesman & Daniels [2000] de acrescentar o sufixo “Mgr”, que representa a abreviação de “Manager”.

Componente	**Descrição**
MessageMgr	Oferece suporte à construção de mensagens. Este componente interage com o canal de comunicação para a transmissão dos dados. Para compor a mensagem podem ser utilizadas várias mídias (pelo menos uma), associando o MessageMgr aos componentes TextualMediaMgr, VideoMediaMgr, AudioMediaMgr ou PictorialMediaMgr. O componente também possibilita o anexo de arquivos na mensagem e seu envio por correio eletrônico.
TextualMediaMgr	Utilizado para mensagens com corpo textual. É possível configurar o tamanho do texto e o vocabulário disponível, através de filtros.
VideoMediaMgr	Utilizado para o tratamento de vídeo. Podem ser configuradas as taxas de captura e de envio de dados.
AudioMediaMgr	Utilizado para o tratamento de áudio. Podem ser configurados a taxa de captura e de envio de dados e o volume.
PictorialMediaMgr	Utilizado para incorporar imagens como parte da comunicação.
DiscreteChannelMgr	Utilizado para transmitir mensagens através de blocos de informação. Pode ser configurado para modo síncrono ou assíncrono, que influencia a maneira como as mensagens são tratadas; push ou pull, que define como a mensagem é entregue ao participante; e o atraso na entrega, que é utilizado para regular a interação em uma ferramenta síncrona [Pimentel et al., 2005].
ContinuousChannelMgr	Utilizado para transmissão contínua de informações.
MetaInformationMgr	Gerencia as meta-informações associadas às mensagens.
CategorizationMgr	Gerencia a categorização de mensagens.
DialogStructureMgr	Gerencia as inter-relações entre as mensagens, que podem ser na forma de lista, árvore ou grafo.
ConversationPathsMgr	Possibilita a utilização de uma máquina de estados para definir os caminhos disponíveis na conversação.
CommitmentMgr	Possibilita o gerenciamento de compromissos negociados durante a comunicação a partir de clichês de conversação.

Tabela 4.2. Componentes de comunicação do Collaboration Component Kit

Cada componente apresenta interfaces fornecidas e requeridas. A Figura 4.8 ilustra o componente de categorização de mensagens (CategorizationMgr). Este componente apresenta a interface ICategorizationMgr, que oferece os serviços relativos à utilização da categorização de mensagens, como atribuir categoria, modificar categoria, listar objetos de uma determinada categoria, etc.; ICategorizationMgrConfig, que oferece os serviços de configuração do componente, como manutenção do conjunto de categorias, estabelecimento da categoria default, etc.; Category, que representa a categoria em si; e ICategorizableObj, que é implementada no serviço colaborativo pelo respectivo objeto a ser categorizado.
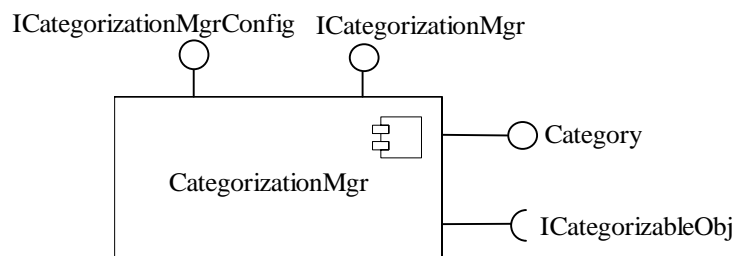
Figura 4.8. Componente de categorização de mensagens

A Tabela 4.3 apresenta os componentes de coordenação do Collaboration Component Kit. Estes componentes objetivam oferecer suporte computacional à organização do grupo, lidando com o acesso dos participantes, com as tarefas atribuídas e com os recursos alocados. O componente AssessmentMgr lida com a avaliação de mensagens; o RoleMgr, com a gestão dos papéis dos participantes; o PermissionMgr, com as permissões; o ParticipantMgr, com os participantes do grupo; o GroupMgr, com a criação de grupos e subgrupos; o SessionMgr, com a gerência das sessões; o FloorControlMgr, com as técnicas de conversação e políticas de acesso; o TaskMgr, com as tarefas e atividades; e o CompetencyMgr, com a gestão das competências dos participantes do grupo. Também são tratadas, através dos componentes AwarenessMgr e AvailabilityMgr, as informações de percepção voltadas para o feedback e feedthrough. O componente NotificationMgr é responsável pelo envio das notificações aos participantes.

Componente	Descrição
AssessmentMgr	Oferece suporte à avaliação qualitativa, possibilitando a associação de notas e conceitos aos objetos compartilhados.
RoleMgr	Cuida do gerenciamento dos papéis atribuídos aos participantes.
PermissionMgr	Gerencia as permissões dos participantes, que podem ser relativas ao papel ou ao indivíduo.
ParticipantMgr	Gerencia os participantes dos grupos.
GroupMgr	Possibilita formar grupos e subgrupos com os participantes.
SessionMgr	Gerencia a sessão de colaboração, incluindo mecanismos para convite e controle de acesso.
FloorControlMgr	Gerencia a ordem de participação. Podem ser alocadas várias políticas de acesso.
TaskMgr	Possibilita o gerenciamento das tarefas do grupo.
AwarenessMgr	Gerencia as informações de percepção em geral, registrando e operando sobre os eventos ocorridos no serviço.
CompetencyMgr	Oferece funcionalidade relativa à gestão das competências dos participantes.
AvailabilityMgr	Possibilita que o participante configure sua disponibilidade.
NotificationMg	Gerencia o envio das notificações aos participantes.

Tabela 4.3. Componentes de coordenação do Collaboration Component Kit

A Tabela 4.4 apresenta os componentes de cooperação do Collaboration Component Kit. Os componentes de cooperação lidam com o registro, a recuperação e o compartilhamento dos objetos. O CooperationObjMgr provê os mecanismos de registro e de concorrência de acesso aos objetos compartilhados. O SearchMgr provê mecanismos de busca. O VersionMgr provê controle de versões, possibilitando a recuperação de edições anteriores e a comparação dos objetos. O StatisticalAnalysisMgr provê funcionalidades para análise estatísticas dos objetos. O RankingMgr possibilita a atribuição de uma avaliação e uma classificação aos objetos, no intuito de ranqueá-los. O RecommendationMgr fornece um mecanismo de recomendação para objetos. O ActionLogMgr oferece recursos para registro das operações realizadas nos objetos. O AccessRegistrationMgr possibilita registrar os acessos que cada participante fez nos objetos, de modo a identificar os objetos já acessados e os novos. O TrashBinMgr oferece uma lixeira para manter temporariamente os objetos removidos, antes da eliminação definitiva.

Componente	Descrição
CooperationObjMgr	Provê mecanismos de compartilhamento de objetos.
SearchMgr	Provê mecanismos de busca para os objetos compartilhados.
VersionMgr	Controla versões dos objetos, possibilitando recuperar uma versão antiga, comparar alterações, entre outras funcionalidades.
StatisticalAnalysisMgr	Oferece funcionalidades de análise estatística dos objetos compartilhados.
RankingMgr	Gerencia a avaliação e a classificação dos objetos compartilhados.
RecommendationMgr	Recomenda objetos para os participantes ou para o grupo a partir das características dos objetos, dos participantes e das tarefas.
ActionLogMgr	Possibilita registrar o histórico de ações no serviço. Pode ser configurado o nível e o tipo de log a ser gerado, possibilitando fazer auditoria, acompanhar o uso, etc.
AccessRegistrationMgr	Registra os acessos dos participantes para cada objeto, possibilitando um controle do que já foi visitado.
TrashBinMgr	Implementa uma lixeira utilizada para armazenar objetos removidos.

Tabela 4.4. Componentes de cooperação do Collaboration Component Kit

Os componentes do Collaboration Component Kit são utilizados na composição dos serviços colaborativos. O conjunto de componentes é iterativamente refinado embasado na realimentação obtida pelo reuso no desenvolvimento das aplicações e pela experimentação das diversas configurações no suporte às características dos grupos e tarefas envolvidos.

Para implementação dos componentes não foram usados os mecanismos para geração de código disponibilizados pelo ambiente Odyssey [Oliveira et al., 2005], pois o modelo de componentes e a infra-estrutura de execução adotados nesta tese possuem especificidades distintas das disponíveis no ambiente. A arquitetura técnica é tratada em mais detalhes na dissertação de Barreto [2006].

4.3. A Arquitetura de Aplicação

Para oferecer suporte ao gerenciamento e à execução dos componentes, são utilizados *component frameworks* [Syzperski, 1997]. Na arquitetura proposta, utiliza-se um *component framework* para cada um dos tipos de componentes propostos, de modo a lidar com as peculiaridades de cada um. No Service Component Framework são acoplados os serviços, oferecendo suporte à montagem do groupware, e no Collaboration Component Framework são acoplados os componentes de colaboração, utilizados na montagem do serviço. A Figura 4.9 ilustra a implantação dos componentes nos *component frameworks*

correspondentes, que estabelecem as condições ambientais para as instâncias dos componentes e oferecem serviços relativos ao ciclo de vida.
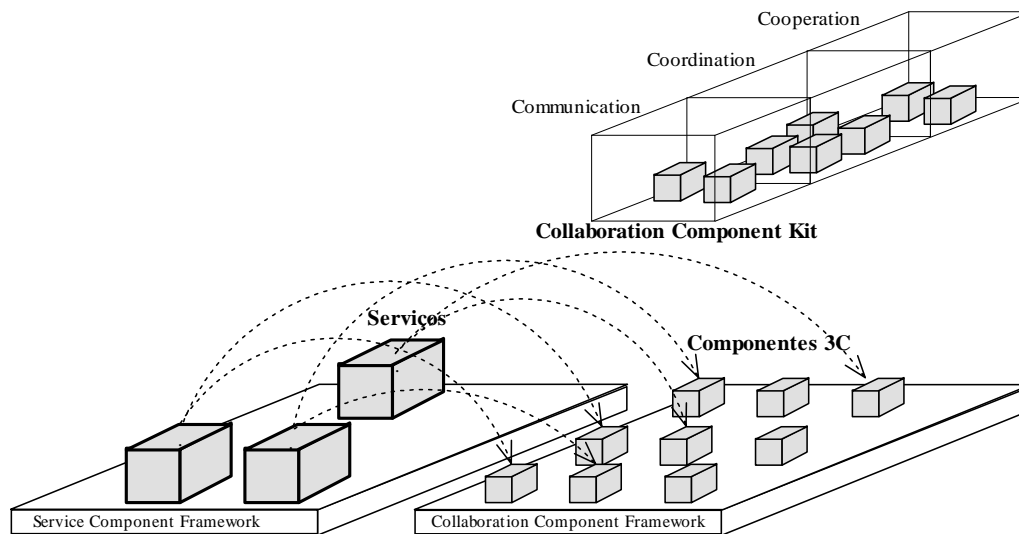
Figura 4.9. Component frameworks com serviços e componentes 3C

Os *component frameworks* são responsáveis por tratar a instalação, remoção, atualização, ativação, desativação, localização, configuração, monitoramento, importação e exportação de componentes. O Service Component Framework gerencia as instâncias dos serviços e a ligação com os componentes de colaboração correspondentes. O Collaboration Component Framework gerencia as instâncias dos componentes de colaboração, que são provenientes do Collaboration Component Kit.

Grande parte das funcionalidades dos *component frameworks* é recorrente e reusável. Um framework pode ser utilizado para a instanciação de uma família de sistemas [Pinto, 2000]. Nesta tese, é utilizado um framework para instanciar os *component frameworks*. Este tipo de framework é chamado de *component framework framework* (CFF) [Szyperski, 1997, p.277]. Um *component framework framework* é visto como um *component framework* de segunda ordem, onde seus componentes são *component frameworks* [Szyperski, 1997, p.276]. Da mesma forma que um componente interage com outros diretamente ou mediado pelo *component framework*, o mesmo pode ser dito dos *component frameworks*, cujo suporte de mais alto nível é o *component framework framework* [Szyperski, 1997, p.277]. A Figura 4.10, estendendo a notação utilizada por Szyperski [1997, p.278], ilustra a arquitetura de aplicação, incluindo o Groupware Component Framework Framework, como o *component framework* de segunda ordem.
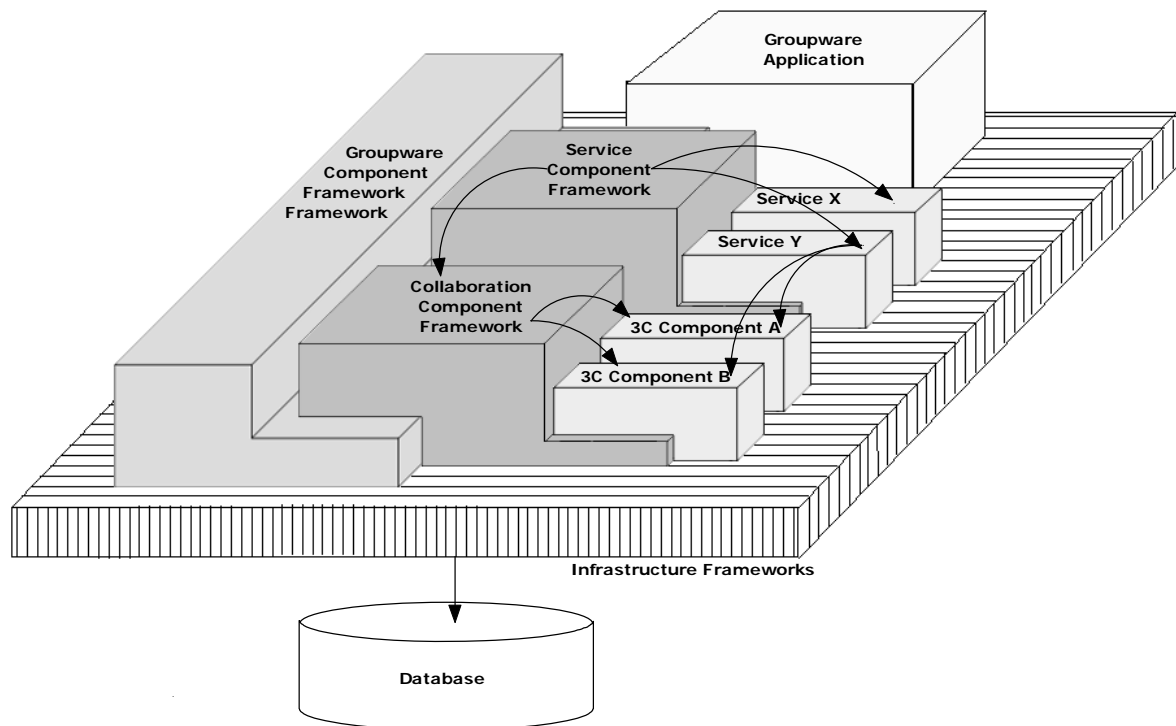
Figura 4.10. A arquitetura de aplicação proposta

A arquitetura adotada segue uma divisão em camadas, onde se distingue a camada de apresentação (não representada na Figura 4.10), que é responsável pela captura e apresentação de dados e pela interação com o usuário; a camada de negócio, que captura o modelo da lógica de negócio do domínio da aplicação; e a camada de infra-estrutura que implementa os serviços técnicos de baixo nível. A divisão em camadas é importante para tratar a complexidade de sistemas componentizados [Szyperski, 1997, p.277].

A mesma infra-estrutura desenvolvida para a camada de negócio pode ser utilizada para mais de uma apresentação, como por exemplo, PDA, desktop implementado em HTML e desktop implementado em Flash (*Rich Internet Application*). Quando os serviços da camada de negócio necessitam de acesso remoto, como por exemplo, em um cliente PDA, são disponibilizados web services que encapsulam a fachada da camada de negócio. Nos demais casos, a apresentação acessa diretamente a fachada do negócio. Esta solução segue os padrões de projeto *Façade* [Gamma et al., 1994], *Data Transfer Object* e *Business Delegate* [Alur et al., 2001].

O ferramental desenvolvido nesta tese instrumenta o desenvolvimento da camada de negócio, implementando os conceitos do modelo 3C de colaboração. A arquitetura de aplicação expressa a estrutura dos componentes do domínio, representando um projeto lógico de alto nível independente da tecnologia de suporte [D'Souza & Wills, 1998]. Já a camada de infra-estrutura é independente do domínio de aplicação e é tratada separadamente na dissertação de Celso Gomes Barreto [2006], que faz parte do consórcio de pesquisa onde esta tese está inserida.

Os *component frameworks* estendem o suporte oferecido pelos frameworks de infra-estrutura. Por exemplo, o container utilizado (JBoss) e os frameworks Spring e Hibernate oferecem suporte a transação declarativa, persistência, integração MVC, injeção de dependências, localização, etc. Na dissertação de Barreto [2006] a arquitetura técnica é tratada em mais detalhes.

O suporte computacional oferecido pelos componentes é reusável para oferecer suporte à colaboração nos diversos grupos da organização, como por exemplo, diretoria, gerência, administração e suporte. Desta forma, em um ambiente educacional, além do suporte à interação em sala de aula, podem ser disponibilizadas ferramentas para mediar a colaboração entre os professores de um curso, de um departamento e de uma instituição. Cada aplicação de groupware implementa sua organização específica.

4.4.
O Modelo de Componentes

Um modelo de componentes define as características dos componentes e de seu ciclo de vida. O modelo de componentes utilizado nesta tese é uma extensão do modelo Java Beans e do modelo oferecido pelo framework Spring. A estes modelos são acrescentadas a definição do ciclo de vida dos componentes, a forma de empacotamento, a sintaxe do arquivo descritor e as regras de interação entre os componentes e os *component frameworks.*

O ciclo de vida de um serviço é ilustrado na Figura 4.11. O componente é posto no sistema de arquivos, em um diretório pré-determinado (d*eployment*). A atividade de *deployment* consiste em implantar um componente para ser instalado

pelo sistema [Szyperski, 2003]. São previstos diretórios específicos para cada um dos Cs do modelo de colaboração. No *deployment*, o componente pode ser customizado editando seu arquivo descritor.
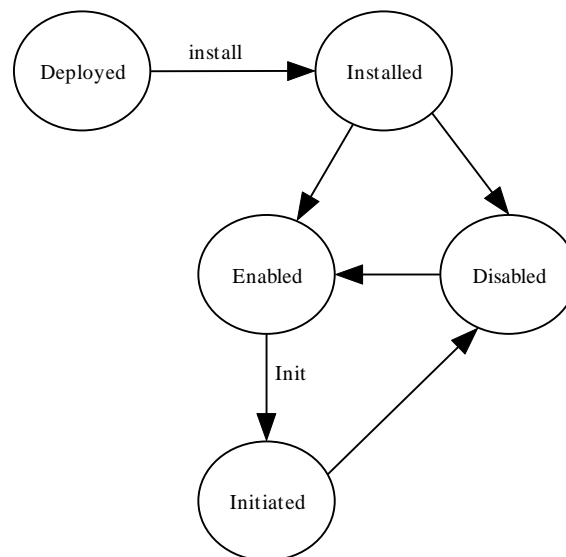
Figura 4.11. Ciclo de vida dos componentes

Para passar do estado de *deployed* para *installed* e se tornar pronto para execução, não há intervenção do desenvolvedor, visto que a atividade de *deployment* não faz parte do desenvolvimento [Szyperski, 2003]. Ao iniciar o servidor de aplicações, o *component framework* verifica a existência de novos serviços e, ao encontrar um novo serviço, executa sua instalação. A instalação do componente consiste em copiar os arquivos, registrar o componente e executar o script de instalação ou de atualização. Neste ponto, as dependências dos componentes e seus estados são lidos e verificados, levando-os para o estado habilitado (*enabled*) ou desabilitado (*disabled*). Os componentes são carregados, inicializados e se tornam prontos para uso.

Um mesmo serviço pode possuir várias instâncias, que são independentes entre si. Por exemplo, no caso do ambiente AulaNet, para cada curso que utiliza um serviço, é criada uma instância do componente[17]. O *Service Component Framework* gerencia as instâncias dos serviços e guarda o estado atual de cada uma, possibilitando a posterior restauração. Na criação de uma nova instância, são utilizados os valores padrões definidos no arquivo descritor.

---

[17] Alguns serviços possuem escopo de turma, sendo necessária uma instância para cada turma.

A instanciação de um novo serviço implica na instanciação dos componentes de colaboração utilizados na composição do serviço. O *Service Component Framework* interage com o *Collaboration Component Framework* para possibilitar a instanciação e a associação entre as instâncias dos componentes. Visando reduzir o acoplamento entre os dois *component frameworks*, é utilizada uma interface que provê um contrato de utilização entre eles.

A instalação e o gerenciamento dos componentes de colaboração seguem um procedimento similar ao descrito para os serviços. Os componentes de colaboração contam com arquivos descritores que definem configurações padrões, utilizadas na instanciação dos componentes. O *Collaboration Component Framework* gerencia a configuração dos componentes de colaboração.

O modelo de componentes define a maneira de tratar os componentes como unidades executáveis de produção, aquisição e instalação independente, definindo a forma de empacotamento do componente [Szyperski, 1997]. Os serviços são empacotados em um arquivo ZIP, com os diretórios CLASSES, DOC, LIBS, SCRIPTS e WEB. Os diretórios CLASSES, WEB e LIBS contêm os arquivos que implementam o componentes e são copiados para os diretórios correspondentes do servidor de aplicações. O padrão J2EE define os diretórios WEB-INF\classes e WEB-INF\libs para receber os arquivos class e jar, e o diretório raiz da aplicação para receber os arquivos que implementam a interface web (html, jsp, js, gif, etc.). O diretório DOCS do empacotamento do serviço disponibiliza a documentação do componente, que inclui a documentação voltada para utilização e para o desenvolvimento e manutenção. Por fim, o diretório SCRIPTS contém os scripts que são executados na instalação, atualização e remoção do serviço. Estes scripts atualizam a base de dados, criam diretórios e alocam os recursos necessários.

O modelo de componentes define também a sintaxe do arquivo descritor. O arquivo descritor é colocado na raiz da estrutura de diretórios do componente. O arquivo descritor é codificado na linguagem XML e segue a estrutura apresentada na Figura 4.12. Optou-se por utilizar a linguagem XML, pois este é o padrão adotado nos frameworks de infra-estrutura utilizados na arquitetura técnica [Barreto et al., 2005]. São definidos no arquivo descritor o identificador, o nome, a descrição, a versão, o tipo e as configurações do serviço e os componentes de

colaboração utilizados, com suas respectivas customizações. No exemplo da figura, o serviço Conferências está utilizando o componente CategorizationMgr.

```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <service>
    <!--                    -->
  - <service-handler>
      <service-name>Conferência</service-name>
      <service-description>Fórum textual de discussão.</service-description>
      <service-type>Asynchronous</service-type>
      <!--              -->
      <service-participation-URL>participation/InitialPage.jsp</service-participation-URL>
      <service-configuration-Initial-URL>config/ConfigPage.jsp</service-configuration-Initial-URL>
    </service-handler>
    <!--                              -->
  - <specific-configuration>
      <allowDesactivation>true</allowDesactivation>
    </specific-configuration>
    <!--                                     -->
  - <collaboration-components>
    - <collabComponent name="categorizationManager">
        <allowNonCategorizedMessage>false</allowNonCategorizedMessage>
      - <defaultCategory>
          <name>Questão</name>
          <descricao>Use a categoria para propor questões</descricao>
        </defaultCategory>
      - <defaultCategory>
          <name>Argumentação</name>
          <descricao>Argumentar um ponto de vista.</descricao>
        </defaultCategory>
      </collabComponent>
    - <collabComponent name="taskManager">
      - <tasks>
        - <task name="Send message" type="collaborative" permissionClass="All">
            <taskComplement>CategorizationManager</taskComplement>
            <taskComplement>ConversationSessionManager</taskComplement>
          </task>
          <task name="Remove message" permissionClass="Mediator" />
          <task name="Alter category" permissionClass="Mediator" />
          <task name="Create conference" permissionClass="Coordinator" />
        </tasks>
      </collabComponent>
    </collaboration-components>
  </service>
```

Figura 4.12. Arquivo descritor de um serviço

Os conectores definidos no modelo de componentes são específicos do domínio e são dependentes do modelo de negócio com o qual os componentes trabalham [Wills, 2001, p.310]. Cada componente disponibiliza um modelo de objetos com os quais os clientes do componente irão lidar. Eventualmente um componente possui uma implementação diferente para um determinado objeto. Nestes casos, adota-se um adaptador que faz a conversão entre eles [Wills, 2001, p.313].

4.5. Instanciação de Groupware

Instrumentado pelo modelo 3C de colaboração, o desenvolvedor modela a aplicação e seus requisitos, e seleciona os serviços e seus componentes de modo a oferecer suporte às necessidades de colaboração. O desenvolvedor seleciona os componentes desejados a partir dos *component kits*, implantando-os nos

component frameworks correspondentes. Se o sistema já estiver em operação, o desenvolvedor identifica os problemas a serem resolvidos a partir da análise da colaboração. As soluções são mapeadas em funcionalidades, que são implementadas pelos componentes de colaboração [Pimentel et al., 2005].

O groupware é montado a partir do levantamento de requisitos, que identifica as necessidades de colaboração e as características do grupo e das tarefas. O trabalho em grupo é modelado e a dinâmica da colaboração é estabelecida. Com base nestas informações, são selecionados os serviços que irão oferecer o suporte computacional à colaboração, definindo a arquitetura da solução. Após a montagem do ambiente, são efetuados os testes e as avaliações, de modo a refinar o ambiente iterativamente. Caso seja necessário modificar a implementação de um serviço existente ou desenvolver um novo, adota-se um ciclo similar ao anterior. Inicia-se com a modelagem das necessidades de colaboração e das características do grupo e das tarefas, no escopo do serviço. A partir destas análises, embasadas pelo modelo 3C, são selecionadas as características relevantes para a solução. Estas características são mapeadas em componentes de colaboração que implementam as funcionalidades requeridas, e o serviço é montado a partir deles.

Caso seja identificada uma necessidade não implementada pelos componentes pré-existentes há três opções: ela pode ser implementada no código da aplicação, de uma forma específica ao domínio em questão; no código do serviço; ou como um novo componente 3C, de modo a disponibilizar a nova funcionalidade para futuro reuso. O novo componente de colaboração é desenvolvido em conformidade com o *Collaboration Component Framework* e com o modelo de componentes estabelecido.

4.6.
Uso de Frameworks de Domínio

Com a evolução do suporte à colaboração, podem ser utilizados frameworks para instanciar famílias de componentes. As funcionalidades de diversos componentes similares são generalizadas e um framework especializável e customizável é implementado. O framework de classes propicia o reuso do código

e fornece ao desenvolvedor mecanismos de especialização, que são utilizados para instanciar famílias de componentes, serviços e aplicações.

No exemplo da Figura 4.13, está representado um framework para geração de diferentes chats. O framework de domínio fundamenta o conhecimento sobre a família de serviços e propicia o reuso do código recorrente. As ligações internas dos componentes são encapsuladas e são oferecidos *hot spots*[18] que ficam disponíveis para o desenvolvedor implementar as especificidades do serviço.
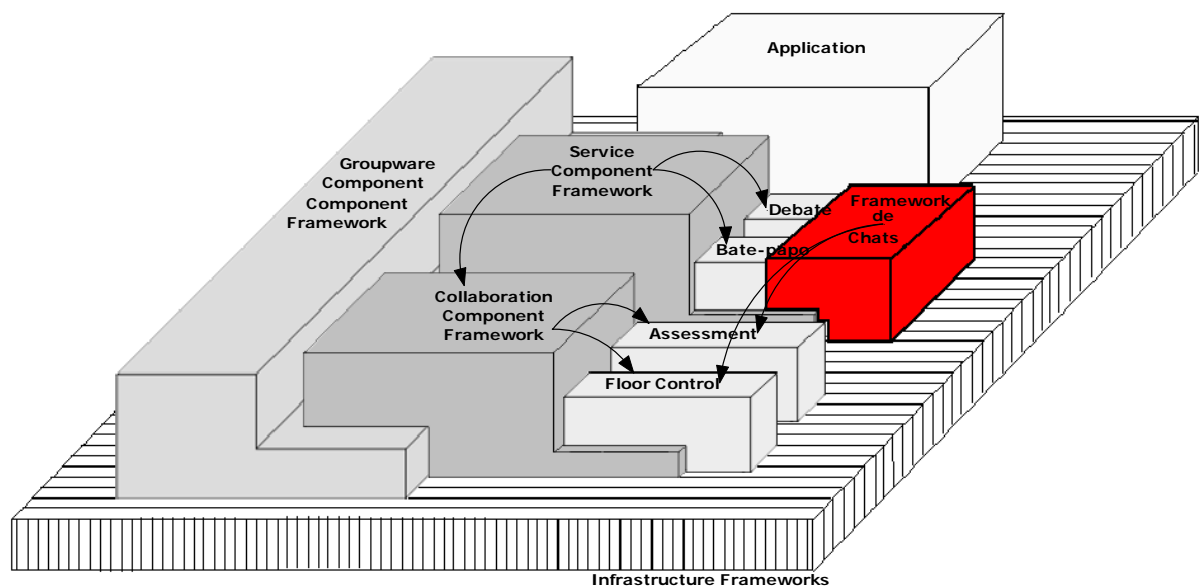
Figura 4.13. Framework de domínio para instanciação de diferentes chats

Podem ser disponibilizados também frameworks para consolidar e gerar uma família de componentes de colaboração, tratando das diversas variações de um mesmo elemento. Por exemplo, ao invés de oferecer ao desenvolvedor diversos componentes de avaliação, pode ser oferecido um framework para instanciar componentes de avaliação, reusando o código recorrente, conforme ilustrado na Figura 4.14.

[18] Para mais detalhes sobre a terminologia associada a frameworks, consulte o Apêndice A.
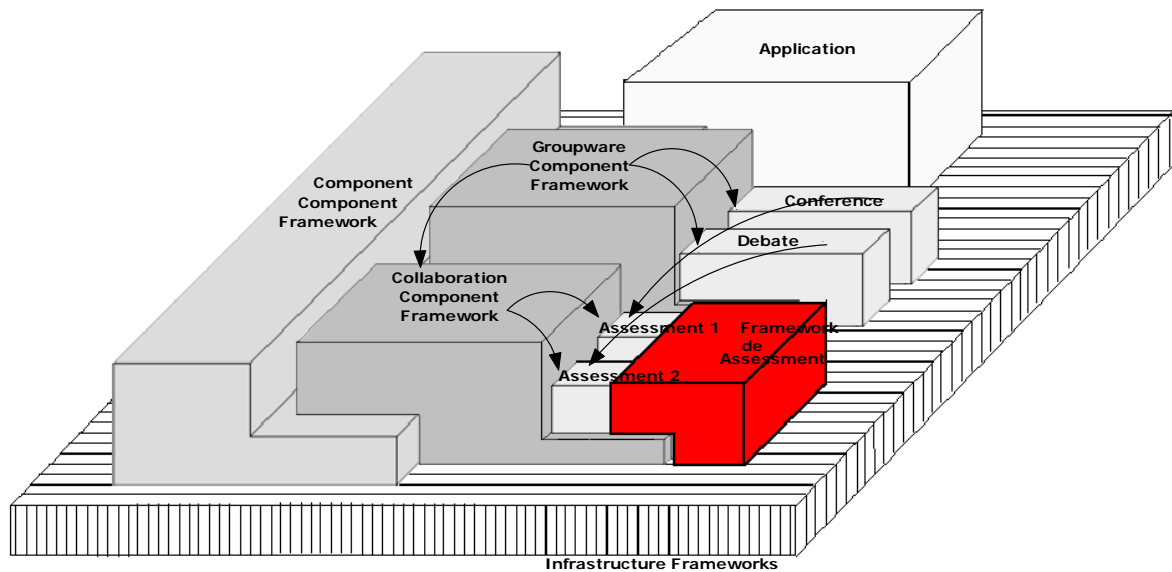
Figura 4.14. Framework de domínio para instanciação de componentes de colaboração voltados para avaliação da participação

A granularidade dos frameworks de domínio é um fator importante a ser considerado. Quanto mais abstrato e abrangente for o framework mais difícil se torna seu uso. Quando mais pontos de extensão, maior a aplicabilidade dentro de um domínio de aplicação, porém sua utilização requer uma longa curva de aprendizado e sua instanciação requer um trabalho maior de customização [Oliveira, 2001]. A granularidade dos componentes e frameworks é iterativamente refinada ao longo do projeto [Gimenes & Huzita, 2005].

Um groupware trata de uma área de aplicação específica, como por exemplo, aprendizagem colaborativa, comércio eletrônico, acompanhamento de projetos, desenvolvimento de software, etc. Frameworks também podem ser utilizados para gerar uma família de groupware de uma mesma área de aplicação, de modo a reusar funcionalidades recorrentes, conforme ilustrado na Figura 4.15. Pode ser desenvolvido, por exemplo, um framework para instanciar ambientes de aprendizagem, a partir do qual é instanciado o AulaNet, entre outros.
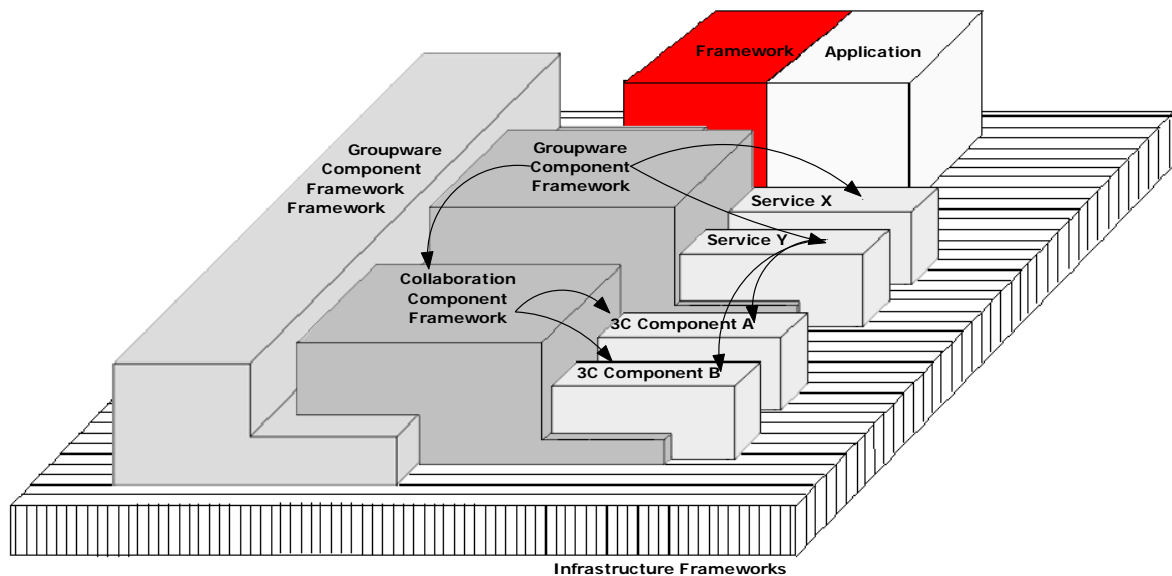
Figura 4.15. Framework de domínio para instanciar uma determinada família de aplicações

Frameworks e componentes são tecnologias complementares [Gimenes & Huzita, 2005; Szyperski, 1997, p.278]. Na abordagem proposta, os frameworks podem ser usados para instanciar os componentes e os ambientes. A abordagem de componentes oferece maior flexibilidade, visto que um framework só oferece variabilidade nos pontos previstos para tal; e maior extensibilidade, visto que pode-se desenvolver e acrescentar novos componentes ou substituir algum em uso, sem impactar a família de aplicações. Entretanto, os frameworks consolidam o conhecimento sobre um subdomínio e possibilitam um maior reuso de código para componentes e ambientes similares.

4.7. Considerações Finais

Groupware é difícil de desenvolver e de manter [Grudin, 1989]. Desenvolver groupware requer diversas áreas de conhecimento, como informática, psicologia, pedagogia, sociologia e cognição. Envolve também várias complexidades técnicas inerentes a sistemas distribuídos e multi-usuário. Um conjunto de componentes interoperáveis fundamentados no modelo 3C de colaboração encapsula parte das dificuldades técnicas e dos aspectos multidisciplinares na implementação dos componentes. Com isto, o desenvolvedor foca sua atenção na investigação dos aspectos específicos de

interação e colaboração, projetando um groupware mais adequado às necessidades do grupo e menos susceptível às falhas de usabilidade.

No desenvolvimento de groupware, os requisitos raramente são claros o suficiente para possibilitar uma especificação precisa antecipada do comportamento do sistema. É difícil prever como um determinado grupo colabora e cada grupo tem características e objetivos consideravelmente distintos [Gutwin & Greenberg, 2000]. Por envolver um grupo, multiplicam-se as possibilidades de interações e aumenta a demanda por sincronismo e concorrência de acesso, o que dificulta a construção de mecanismos de interação adequados e a condução de testes. Usar um conjunto de componentes que são reusados em diversas situações aumenta a confiabilidade e a estabilidade do sistema, além de possibilitar substituir ou reimplementar componentes com impactos limitados no restante do sistema. O desenvolvedor também pode experimentar e prototipar diversas configurações iterativamente para refinar os requisitos do sistema e o suporte à colaboração.

O uso de componentes oferece capacidade de customização e extensão. Oferecer um conjunto de componentes torna desnecessário prever todas as possibilidades de utilização e dar suporte a elas. São oferecidos blocos de granularidade média que o desenvolvedor usa para compor a aplicação [Szyperski, 1997]. Estes blocos também oferecem suporte à pluralidade de plataformas a partir das quais são acessados os sistemas colaborativos. Isto inclui computadores desktop e dispositivos móveis, como celulares e PDAs, que estão cada vez mais com recursos suficientes para incrementar a comunicação, a coordenação e a cooperação do grupo [Filippo et al., 2005].

Software é evolutivo, e groupware é evolutivo por natureza [Tam & Greenberg, 2004]. A composição e as características dos grupos de trabalho se alteram ao longo do tempo, assim como as tarefas executadas. O grupo aprende, surgem afinidades e conflitos entre os membros, entram e saem pessoas, levando o grupo a mudar continuamente. A componentização provê a capacidade de montar e evoluir um ambiente de trabalho específico, selecionando e configurando um conjunto de ferramentas colaborativas específicas para suas necessidades.

Esta pesquisa propõe a estruturação de um sistema colaborativo em componentes que encapsulam as dificuldades técnicas de sistemas distribuídos e

multi-usuário e refletem os conceitos da colaboração, modelados pelo modelo 3C. Este ferramental instrumenta o desenvolvimento da camada de negócio da aplicação, possibilitando trocar a camada de interface de um serviço, mantendo a mesma lógica de aplicação. No contexto deste trabalho, a engenharia de domínio é baseada no modelo 3C de colaboração, de modo a guiar o desenvolvimento de aplicações colaborativas. A transição entre as atividades do desenvolvimento e o mapeamento dos conceitos de análise para as estruturas do código é estreitado, facilitando o desenvolvimento iterativo e a posterior manutenção da aplicação.

“Sem uma arquitetura adequada, a construção de groupware e sistemas interativos em geral é difícil de obter, o software resultante é difícil de manter e o refinamento iterativo é dificultado” [Calvary et al., 1997]. Uma arquitetura componentizada possibilita que componentes sejam selecionados para montar um groupware voltado aos interesses específicos de um grupo. Os componentes são customizados e combinados na medida da necessidade, tendo em mente futuras manutenções. O uso desta abordagem propicia a prototipação e a experimentação, que são fundamentais em CSCW, visto que ainda são escassos e pouco documentados os casos de sucesso. A prototipação deve ser rápida, pois ocorre enquanto o grupo trabalha, para se obter um feedback oportuno e proceder com os ajustes. O uso de componentes auxilia a adaptação dinâmica do ambiente e do suporte à colaboração através da recomposição e reconfiguração do sistema.

Entretanto, vale ressaltar que a solução proposta não elimina a necessidade de um desenvolvedor consciente e conhecedor do assunto em questão, pois não basta sair ligando os componentes aleatoriamente para produzir um sistema colaborativo eficaz. O uso de um processo sistematizado ajuda a reduzir estas dificuldades, ao instrumentar o desenvolvedor com catálogos de problemas, soluções, elementos e componentes, bem como as atividades a serem executadas e os artefatos a serem produzidos para obter o produto final.

Um processo é um conjunto de políticas, tecnologias, procedimentos, artefatos e estruturas organizacionais utilizado para conceber, desenvolver, implantar e manter um produto de software [Fuggetta, 2000]. No consórcio de pesquisa no qual esta tese está inserida, Mariano Pimentel [2006] propõe um processo de desenvolvimento de groupware que leva em consideração a arquitetura e a abordagem propostas nesta tese. Este processo instrumenta e guia o

desenvolvedor, que passa a contar com uma abordagem sistematizada para o desenvolvimento incremental de groupware baseado no modelo 3C [Pimentel et al., 2005]. O processo estabelece a ligação entre os problemas e soluções, bem como o mapeamento para as funcionalidades e componentes correspondentes. O processo define as atividades e artefatos que são gerados durante o desenvolvimento e instrumenta o desenvolvedor para que a partir dos sintomas possa identificar quais elementos precisam ser refinados. O processo estabelece técnicas e abordagens para avaliar uma ferramenta, com o objetivo de identificar problemas e proceder em novos ciclos de desenvolvimento.

5
Estudos de Caso

Neste capítulo são analisados estudos de caso visando avaliar a abordagem proposta nesta tese. É descrita a instanciação da solução proposta para o desenvolvimento do AulaNet, apresentando a evolução da arquitetura do ambiente e dos serviços Conferências e Debate. Além disto, para prover indícios da aplicabilidade da solução em outras situações e problemas, são discutidos opiniões de especialistas, a utilização por não-especialistas e o mapeamento do ambiente TelEduc.

5.1.
Estudos de Caso no Ambiente AulaNet

O AulaNet iniciou-se como uma ferramenta para apoiar um curso a distância pela web e evoluiu para um ambiente de ensino-aprendizagem para vários cursos. Sua primeira versão foi desenvolvida utilizando a linguagem Lua [Ierusalimschy et al., 1996]. No ano de 1998 iniciou-se a parceria entre a PUC-Rio e a empresa EduWeb para a distribuição e customização do ambiente, e o AulaNet 1.2 foi lançado para o mercado [Lucena et al., 1998].

Em 1999, percebendo uma desestruturação e limitações do código da aplicação, que evoluiu muito rapidamente, resolveu-se re-implementar o AulaNet. Decidiu-se utilizar a linguagem de programação Java, de modo a melhorar a integração com o mercado de desenvolvedores e aproveitar a robustez, portabilidade e recursos da plataforma. A versão 2.0 do AulaNet foi lançada em Dezembro de 2001.

A versão 2.0 do AulaNet foi desenvolvida em uma arquitetura cliente-servidor na web com a tecnologia Scriba [Blois et al., 1999], que através de um *servlet* intermedeia a comunicação do cliente com o servidor. O Scriba oferece uma linguagem própria de script que é embutida nos arquivos HTML. Esta linguagem possibilita, entre outras facilidades, acesso a banco de dados, definição

de variáveis para armazenamento temporário de dados e chamadas a classes implementadas em Java. As classes agrupam as funções mais complexas e específicas da aplicação. A Figura 5.1 ilustra a arquitetura do AulaNet 2.0.
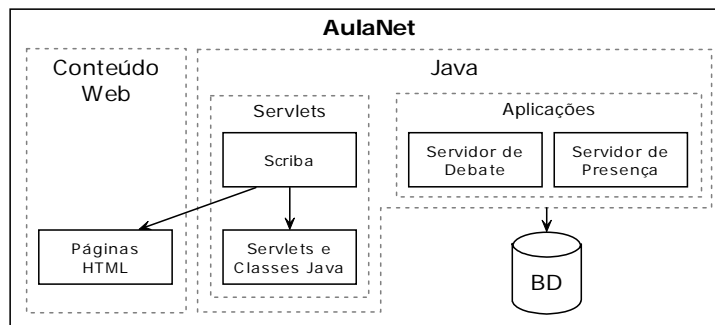
Figura 5.1. Arquitetura do AulaNet 2.0

Quando o Scriba foi desenvolvido, a Sun Microsystems ainda não havia disponibilizado a tecnologia Java Server Pages (JSP), que possui o mesmo propósito do Scriba. O Scriba foi criado especialmente para servir de base para o ambiente AulaNet. O uso de tecnologias caseiras, como o Scriba, para resolver problemas já bem resolvidos por outras tecnologias, dificulta a evolução do ambiente. Cada novo integrante da equipe de desenvolvimento do LES ou da EduWeb necessita aprender a tecnologia. Passa a ser necessário também estender a linguagem para acompanhar os avanços e promover a interoperabilidade do ambiente. A linguagem passou a ser um ponto a mais para documentar e manter.

Outro problema relacionado à utilização do Scriba é a disseminação de código por arquivos HTML e classes Java. Na arquitetura atual do AulaNet os arquivos HTML contêm comandos Scriba, que geram dinamicamente as páginas, recorrendo a classes implementadas em Java para as funções mais complexas. Esta disseminação de código entre as diferentes linguagens e arquivos dificulta a manutenção do ambiente.

Classes Java implementam funcionalidades específicas de cada página. Apesar de implementadas em uma linguagem orientada a objetos, estas classes comportam-se como biblioteca de funções, assemelhando-se ao paradigma de programação procedural [Cousineau & Mauny, 1998], com pouco reuso. O acesso aos dados do sistema é espalhado pelo código, de modo que quando ocorre alguma alteração no banco de dados é necessário buscar e atualizar todas as classes e arquivos.

Sob a coordenação dos professores Carlos José Pereira de Lucena e Hugo Fuks, o AulaNet vem sendo desenvolvido por prototipação, por alunos do doutorado, mestrado e graduação da PUC-Rio que, além de mantê-lo, usam-no em suas teses, dissertações e monografias, implementando e testando os conceitos de seus trabalhos. O AulaNet cresceu e suas funcionalidades foram implementadas na medida da necessidade. Mais de 20 alunos atuaram diretamente no desenvolvimento do AulaNet, em períodos distintos. A rotatividade de desenvolvedores e de enfoques levou a um código despadronizado.

Por estes diversos fatores, tornou-se custoso realizar alterações no AulaNet, desenvolver novos serviços e evoluir os existentes. Além disto, vem sendo custosa a integração de novos membros à equipe de desenvolvimento.

5.1.1. O AulaNet 3.0

A nova versão do AulaNet está sendo completamente reescrita, utilizando a abordagem proposta nesta tese. São utilizados componentes concebidos com base no modelo 3C e que encapsulam um conjunto coeso de dados e funções. Os *component frameworks* encapsulam e oferecem serviços de baixo nível, dando suporte ao desenvolvimento e manutenção de groupware. Nesta seção são discutidos alguns requisitos da nova versão do ambiente com relação à abordagem proposta e à arquitetura utilizada. Os requisitos foram consolidados e discutidos em reuniões das equipes de desenvolvedores do grupo AulaNet da PUC-Rio e da EduWeb. Foram realizadas 15 reuniões de Janeiro a Junho de 2005. Os requisitos têm origem na experiência de 8 anos de desenvolvimento do AulaNet e nas demandas do meio corporativo atendido pela EduWeb.

Um exemplo de demanda recorrente no meio empresarial é a adaptação da estrutura hierárquica e de papéis para cada empresa e instituição. A versão 2.0 do AulaNet possui uma organização fixa baseada em Instituições e Departamentos, com os seguintes papéis: administrador, coordenador, docente co-autor, mediador e aprendiz. Entretanto, na prática, as instituições possuem diferentes níveis de administração, como departamentos, unidades, filiais, regiões, instituições, núcleos, etc. e papéis com funcionalidades específicas, como chefe de

departamento, diretor, coordenador, consultor, gerente, etc. Os cursos também são agrupados de diferentes maneiras: currículos, itinerários, grades, disciplinas, etc. e em cada curso há papéis específicos, como por exemplo, monitor, assistente, ouvinte, supervisor, conteudista, mentor, auditor, etc. A utilização de componentes para gerenciar papéis, grupos hierárquicos e cursos favorece a flexibilização destas estruturas. No entanto, para suavizar a transição e possibilitar a comparação do AulaNet 2.0 com o 3.0, decidiu-se implementar componentes que de início capturem as mesmas funções disponibilizadas na versão 2.0 do ambiente, já prevendo sua posterior substituição por componentes mais robustos que atendam às extensões dos requisitos. Os componentes 3C de coordenação GroupMgr, RoleMgr e PermissionMgr endereçam estes requisitos.

Outra flexibilidade requerida pelas instituições é na definição das informações sobre participantes e cursos. Cada instituição tem necessidades específicas de informações com relação a estas entidades, como por exemplo, nome, endereço, telefone, idade, sexo, foto, etc. para participante; e carga horária, custo, fornecedor, ramo de aplicação, cronograma, etc. para curso. As informações necessárias dependem do escopo do ambiente e da política vigente na instituição. A adoção de componentes para gerenciar estas entidades favorece a customização das informações necessárias. Os componentes 3C de coordenação GroupMgr e ParticipantMgr provêem estas funcionalidades. O componente GroupMgr é utilizado para implementar os cursos do ambiente, que são tratados como grupos de participantes associados a serviços colaborativos.

Na solução adotada, para aumentar a modularidade dos serviços do ambiente e reusar a infra-estrutura de execução, também são tratados como componentes os serviços administrativos (matricular aprendizes, gerenciar turmas, configurar o ambiente, etc.), os serviços para os visitantes (FAQ, contato com administrador, notificação de erro, listagem de cursos, etc.) e os serviços para participantes (criar curso, matricular, configurar perfil, etc.). As quatro famílias de serviços do AulaNet são ilustradas na Figura 5.2. Os serviços colaborativos são utilizados pelos docentes e aprendizes para dar suporte às atividades de aprendizagem colaborativa; os serviços administrativos são utilizados pelo administrador do sistema para gerenciar e configurar as funcionalidades e dados do servidor; os serviços de participantes são utilizados pelos usuários

autenticados; e os serviços de visitantes são disponibilizados para os usuários não autenticados. O AulaNet provê um conjunto padrão de serviços adaptável para cada servidor, de modo que cada instituição seleciona um conjunto de serviços específicos para suas necessidades.

Figura 5.2. Serviços de colaboração, administrativos, para participantes e para visitantes do ambiente AulaNet

A classificação do serviço é feita na sua implantação (*deployment*) no Service Component Framework. Os serviços de administração, de visitantes e de participantes utilizam a infra-estrutura de execução provida pelos *component frameworks* e componentes 3C. Por exemplo, o serviço Contato com Administrador, que é disponibilizado para os visitantes, utiliza componentes de comunicação para o envio e recebimento de mensagens (MessageMgr, DiscreteChannelMgr), componentes de coordenação para o controle de papéis e permissões (RoleMgr e PermissionMgr) e componentes de cooperação para registro e busca (CooperationObjMgr e SearchMgr). Mesmo serviços que não pressuponham interações diretas entre participantes utilizam componentes 3C, como o RoleMgr, PermissonMgr, CooperationObjMgr, entre outros.

A utilização de componentes e *component frameworks* também possibilita que um serviço colaborativo seja utilizado em um contexto que não seja o do

curso. Por exemplo, o serviço Conferências pode ser utilizado no menu de participantes, para que todos usuários do ambiente discutam assuntos gerais. O *component framework* também possibilita a instalação de um mesmo serviço mais de uma vez, com nomes e identificadores diferentes.

Com a estrutura de grupos que agregam serviços e participantes, pode ser oferecido suporte à colaboração nos diversos níveis organizacionais da instituição, como por exemplo, membros da diretoria, das coordenações de curso, do conselho acadêmico, etc. Também é possível oferecer suporte específico para subgrupos, como mediadores de uma turma, aprendizes que realizam uma tarefa colaborativa, co-autores de um curso, entre outros. Por alterarem a funcionalidade e a organização do AulaNet 2.0 do ponto de vista do usuário, estas características serão implantadas em futuras versões do ambiente.

As instituições que utilizam o ambiente possuem necessidades específicas de interface com o usuário. A estruturação em camadas da nova arquitetura do AulaNet possibilita utilizar interfaces distintas para as instituições, reusando a lógica do suporte à colaboração. Está prevista para a nova versão do ambiente a utilização de pacotes de elementos de interface com o usuário (*skins*) que alteram a apresentação como um todo. A separação em camadas também possibilita utilizar tecnologias de interfaces rica (RIA – *Rich Internet Application*) quando houver recursos computacionais e de transmissão de dados apropriados.

A separação em camadas também favorece a utilização de dispositivos móveis, como PDAs e celulares, provendo um maior suporte ao desenvolvimento do AulaNetM [Filippo et al., 2005]. Quando é necessário executar código Java no lado cliente, são utilizados web services que disponibilizam os serviços da fachada da camada de negócio da aplicação.

Os *component frameworks* utilizados encapsulam várias funcionalidades de baixo nível que implementam necessidades estabelecidas para a versão 3.0 do AulaNet. Por exemplo, passou-se a utilizar sessão para registrar dados temporários do usuário e aproveitar o mecanismo de autenticação oferecido pelo container, que melhora sensivelmente a segurança do sistema, possibilitando o controle de permissões por página. São utilizadas também transações, de modo a evitar a inconsistência de dados. A infra-estrutura disponibiliza também recursos

para gerar logs de ações no ambiente, de modo a possibilitar a auditoria e a volta a estágios anteriores; e a importação e exportação de dados.

Para facilitar a manipulação dos serviços pelo administrador do ambiente, cada serviço é empacotado em um arquivo compactado e, para ser instalado, é copiado a um determinado diretório. O administrador instala, atualiza ou remove serviços, incluindo, substituindo e excluindo arquivos. O administrador dispõe de um serviço chamado Gerenciador de Serviços, onde o nome, a versão e a data da última utilização dos serviços são apresentados e ativa-se e desativa-se serviços, possibilitando experimentar diversas configurações. As customizações dos componentes são feitas nos descritores dos serviços, que são editáveis em qualquer editor de texto. Ao ser inicializado, o *component framework* lê os dados do sistema de arquivos e detecta mudanças no conjunto de componentes e instala, atualiza ou remove os serviços correspondentes. O *component framework* também verifica eventuais dependências de componentes e versões.

A padronização dos conectores e interfaces entre os serviços e o *component framework* possibilita o desenvolvimento de novos serviços, mesmo sem conhecer a implementação interna do AulaNet. Instituições podem desenvolver novos serviços ou integrar serviços já existentes ao ambiente, direcionando seus esforços para o desenvolvimento do apoio educacional, como laboratórios e simuladores, deixando para o AulaNet o suporte ao gerenciamento de cursos e participantes.

5.1.2.
A Arquitetura do AulaNet 3.0

A Figura 5.3 ilustra a arquitetura apresentada no capítulo anterior instanciada para o ambiente AulaNet. A aplicação trata a parte específica do domínio e a integração entre os serviços escolhidos. São tratados, por exemplo, os cursos e o mapeamento das turmas para os grupos de colaboração. Os serviços do ambiente são plugados no Service Component Framework e utilizam os componentes 3C presentes no Collaboration Component Framework.
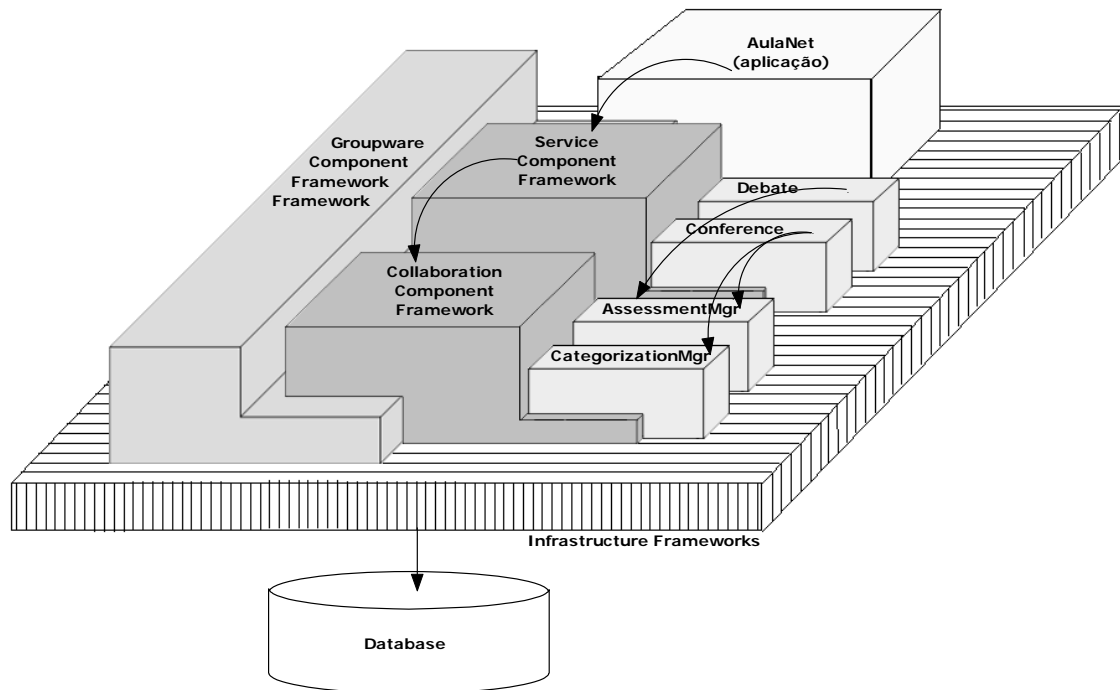
Figura 5.3. Arquitetura instanciada para o ambiente AulaNet (a título de clareza, somente alguns serviços e componentes são apresentados)

Na próxima seção são analisados estudos de caso utilizando a instanciação do AulaNet para apresentar alguns dos conceitos discutidos nesta tese em cenários reais e fictícios de utilização.

5.1.3. Composição no AulaNet 3.0

A solução proposta nesta tese foi utilizada para implementar a nova versão do AulaNet. O mapeamento dos serviços de comunicação do ambiente para os componentes 3C está apresentado na Tabela 5.1.

COMPONENTE 3C	SERVIÇO					
	Conferências	**Correio para Turma**	**Correio para Participante**	**Bate-Papo**	**Debate**	**Mensageiro**
COMUNICAÇÃO						
MessageMgr	X	X	X	X	X	X
TextualMediaMgr					X	
VideoMediaMgr						
AudioMediaMgr						
PictorialMediaMgr						
DiscreteChannelMgr	X	X	X	X	X	X
ContinuousChannelMgr						
MetaInformationMgr						
CategorizationMgr	X	X				
DialogStructureMgr	X		X			
ConversationPathsMgr						
CommitmentMgr						
COORDENAÇÃO						
AssessmentMgr	X	X		X	X	
RoleMgr	X	X	X	X	X	X
PermissionMgr	X	X	X	X	X	X
ParticipantMgr	X	X	X	X	X	X
GroupMgr						
SessionMgr	X	X		X	X	
FloorControlMgr					X	
TaskMgr						
AwarenessMgr	X	X	X	X	X	X
CompetencyMgr	X	X			X	
AvailabilityMgr						
NotificationMgr	X	X				
COOPERAÇÃO						
CooperationObjMgr	X	X	X	X	X	
SearchMgr	X					
VersionMgr						
StatisticalAnalysisMgr	X					
RankingMgr						
RecommendationMgr						
ActionLogMgr	X	X	X	X	X	X
AccessRegistrationMgr	X	X	X			
TrashBinMgr						

Tabela 5.1. Mapeamento dos serviços de comunicação aos componentes 3C

Para avaliar a utilização da abordagem proposta no desenvolvimento de groupware, é ilustrada a montagem, adaptação, substituição, reuso e extensão da solução, enfocando o suporte computacional à colaboração. É abordada a instanciação do AulaNet a partir dos serviços, e a instanciação destes a partir dos componentes 3C. O curso TIAE, apresentado no Capítulo 3, é utilizado como um cenário real, de onde são mapeadas as necessidades de colaboração para arranjos de componentes.

A dinâmica do curso TIAE, apresentada no Capítulo 3 e na Figura 5.4, prevê as seguintes atividades: apresentação, estudo individual dos conteúdos do curso, argumentação e alinhamento de idéias sobre os tópicos semanais, produção colaborativa de conteúdo multimídia interativo, revisão dos conteúdos em pares,

submissão dos conteúdos e esclarecimento de dúvidas. Ao projetar o suporte computacional à colaboração no curso, são selecionados serviços para cada uma das atividades.
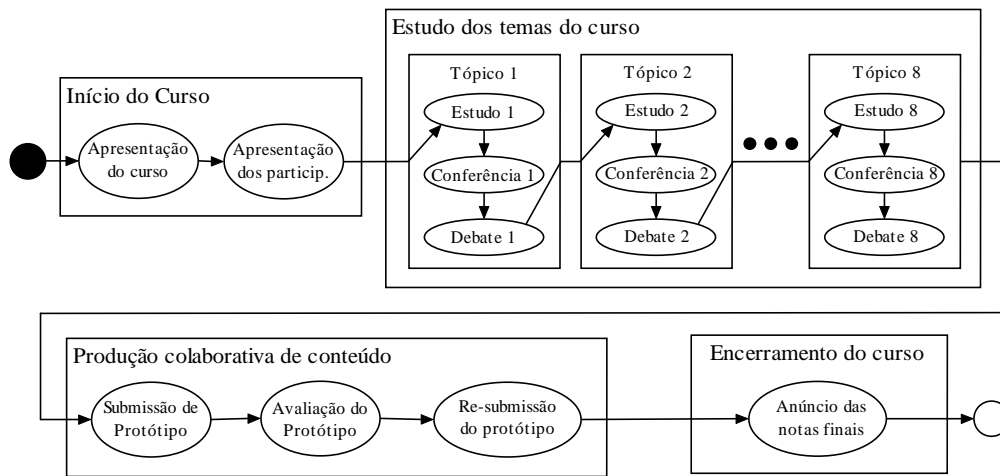
Figura 5.4. Fluxo de atividades no curso TIAE

A Tabela 5.2 apresenta o mapeamento entre as atividades e os serviços selecionados. Para a apresentação dos aprendizes é utilizado o serviço Correio para Turma. Este serviço registra e dispara a mensagem de apresentação para a caixa de correio eletrônico de cada participante, o que é importante para alertar os aprendizes de que o curso começou, já que esta é a primeira atividade. Para o estudo individual dos tópicos do curso são disponibilizados conteúdos através dos serviços Aulas, Bibliografia e Webliografia. A discussão sobre estes conteúdos é dividia em duas etapas: argumentação nas Conferências e alinhamento de idéias no Debate, de modo a aproveitar a reflexão e o aprofundamento propiciados pelo primeiro e a velocidade de interação propiciada pelo segundo. Para a produção do conteúdo educacional multimídia não é disponibilizado nenhum serviço do AulaNet, possibilitando aos aprendizes utilizarem os ambientes de autoria com os quais já estejam habituados a lidar. Para a revisão em pares, é utilizado o serviço Conferências, pois deseja-se valorizar a argumentação e o aprofundamento da discussão. O serviço Tarefas oferece as funcionalidades necessárias para gerenciar as submissões dos conteúdos pelos aprendizes. Para dúvidas, são disponibilizados os serviços Correio para Participante, que propicia um canal particular com os mediadores do curso, e Correio para Turma, que é aberto para toda a turma. Este serviço também é utilizado para o anúncio dos resultados finais do curso. O modelo 3C guia a seleção e busca do serviço a ser utilizado para oferecer suporte

computacional à dinâmica estabelecida para a colaboração. A partir da seleção de serviços, o groupware é construído utilizando os componentes correspondentes[19].

Atividade	Serviço
Apresentação	Correio para Turma (comunicação)
Estudo Individual	Aulas, Bibliografia e Webliografia (cooperação)
Argumentação	Conferências (comunicação)
Alinhamento de Idéias	Debate (comunicação)
Produção de conteúdo	-
Revisão em pares	Conferências (comunicação)
Submissão de conteúdo	Tarefas (coordenação)
Dúvidas	Correio para Participante e para Turma (comunicação)
Anúncio do Resultado Final	Correio para Turma (comunicação)

Tabela 5.2. Serviços do AulaNet utilizados no curso TIAE

A partir da realimentação obtida com o uso dos serviços, iterativamente ajusta-se o suporte computacional à colaboração. Caso haja alteração na dinâmica do curso, recompõe-se o ambiente acrescentando, substituindo ou retirando serviços. Por exemplo, se o coordenador julgar que é necessário avaliar o conhecimento sobre o tópico semanal através de testes, pode alterar a dinâmica do curso para incluir esta atividade, conforme ilustrado no fluxo da Figura 5.5. Sendo a atividade de avaliação uma atividade de coordenação, é buscado nos serviços de coordenação um serviço que ofereça suporte computacional a esta atividade. No caso do AulaNet pode ser utilizado o serviço Exames, que possibilita a criação de questionários, com questões de múltipla escolha, verdadeiro ou falso, discursiva, etc. classificadas de acordo com a taxionomia de Bloom [1956]. O novo serviço é incorporado ao ambiente e passa a ser utilizado no curso. Se o serviço não atender, são buscados, no próprio kit ou em outros repositórios, componentes que implementem questionários ou outras formas de avaliação.

[19] Como o AulaNet oferece suporte a mais de um curso, não é viável possibilitar que cada coordenador instale ou remova serviços do ambiente, pois isto iria atingir todos os cursos hospedados. Por isto, é oferecida ao coordenador do curso a capacidade de seleção e ativação dos serviços no contexto de seu curso. Os serviços disponíveis são aqueles instalados pelo administrador, que monta um ambiente de acordo com as necessidades da instituição.
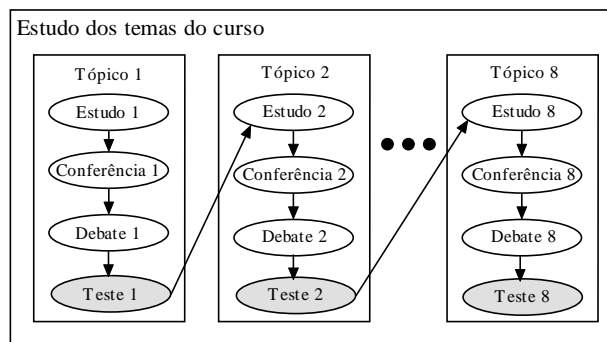
Figura 5.5. Alteração no fluxo de atividades no curso TIAE para incluir uma avaliação

Além da inclusão de novos serviços, a realimentação obtida com a utilização do ambiente pode levar à substituição de serviços existentes. Por exemplo, se os coordenadores do TIAE julgarem que os aprendizes estão tendo dificuldades com o serviço Debate, podem substituí-lo pelo Bate-Papo, que apresenta uma interface mais simples e com menos funcionalidades, conforme pode ser observado na Figura 5.6. Também pode iniciar com o serviço mais simples e quando os aprendizes se ambientarem, substituí-lo pelo mais robusto. Um problema desta substituição é que as sessões, avaliações, participações, etc. ainda ficam registradas para o serviço anterior. Se os serviços forem compatíveis utiliza-se o recurso de importação/exportação do ambiente para transferir os dados.
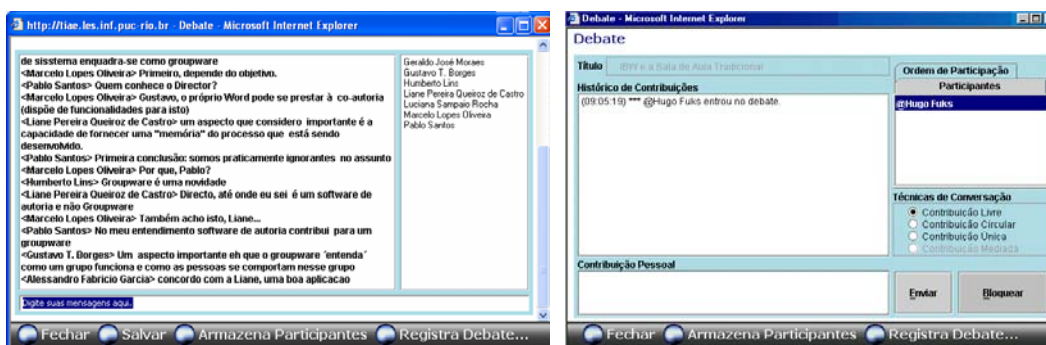
Figura 5.6. Serviços Bate-Papo e Debate

Algumas modificações são resolvidas com customizações nos serviços, não necessitando de sua troca. Por exemplo, para habilitar a possibilidade de desativação da Conferência, basta alterar a propriedade correspondente no arquivo descritor do componente, que é apresentado na Figura 5.7. Dependendo da dinâmica estabelecida para o uso do serviço pode-se ajustá-lo, configurando suas funcionalidades.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <service>
   <!-- Service Descriptor: Information for the Component Framework -->
 - <service-descriptor>
     <!-- The component Id is the folder name -->
     <service-name>Conferências</service-name>
     <service-description>Fórum de discussão assíncrono.</service-description>
     <service-type>Asynchronous</service-type>
     <!-- Service View -->
     <service-participation-URL>/aulanet3/loadConference.do</service-participation-URL>
     <service-configuration-URL />
     <!-- Service Model -->
     <ServiceMgrClass>services.conference.model.ConferenceMgr</ServiceMgrClass>
     <ServiceInstanceClass>services.conference.model.ConferenceInstance</ServiceInstanceClass>
   </service-descriptor>
   <!-- Component specific configuration -->
 - <specific-configuration>
     <allowDesactivation>true</allowDesactivation>
   </specific-configuration>
   <!-- Collaboration Components Used -->
 - <collaboration-components>
     <collabComponent id="collabcomponent.coop.cooperationObjMgr" />
   - <collabComponent id="collabcomponent.comunic.categorizationMgr">
     - <categories>
       - <Category>
           <name>Seminário</name>
           <description>O seminarista da semana escreve uma mensagem que agrega valor sobre os conteúdos do tema
             da semana. [raiz]</description>
         </Category>
       - <Category>
           <name>Argumentação</name>
```

Figura 5.7. Trecho do arquivo descritor das Conferências

Encapsular os serviços na forma de componentes também possibilita repetir a instalação de um mesmo serviço com configurações e características diferentes para atender a tarefas distintas. Por exemplo, para o curso TIAE o serviço Conferências pode ser duplicado (duplicando o arquivo correspondente na estrutura de diretórios), já que as Conferências são utilizadas para a argumentação sobre os temas semanais e para a avaliação em pares. Cada instalação é nomeada e configurada de forma específica. As sessões de cada atividade são separadas, possibilitando uma maior adequação dos relatórios e estatísticas, precisão nas buscas e adoção de categorias, papéis, permissões e critérios de avaliação diferentes.

Os cenários abordados ilustram que manipulando arquivos no sistema operacional ou alterando propriedades em arquivos textuais é possível alterar a configuração e o suporte à colaboração do ambiente. Caso o ambiente não ofereça um determinado serviço é possível estendê-lo incorporando o novo serviço, sem desestruturar os demais.

Para encapsular uma ferramenta que não foi originalmente desenvolvida para o ambiente, é necessário criar um empacotamento que possibilite instalá-la. É necessário criar o arquivo descritor, os scripts e a estrutura de diretórios definidos no modelo de componentes do AulaNet. Por exemplo, a Figura 5.8 ilustra a

utilização da ferramenta externa de montagem de calendário tCalDate[20], que foi encapsulada para operar a partir do ambiente AulaNet. Como esta ferramenta não mantém uma base paralela de usuários e grupos, não foi necessário adaptar seu código fonte para operar com a base de usuários do AulaNet, tendo sido apenas necessário criar o empacotamento e o descritor. A despadronização visual e a eventual necessidade de adaptação do código fonte da ferramenta, que nem sempre está disponível, são alguns dos problemas enfrentados na adaptação de ferramentas externas.
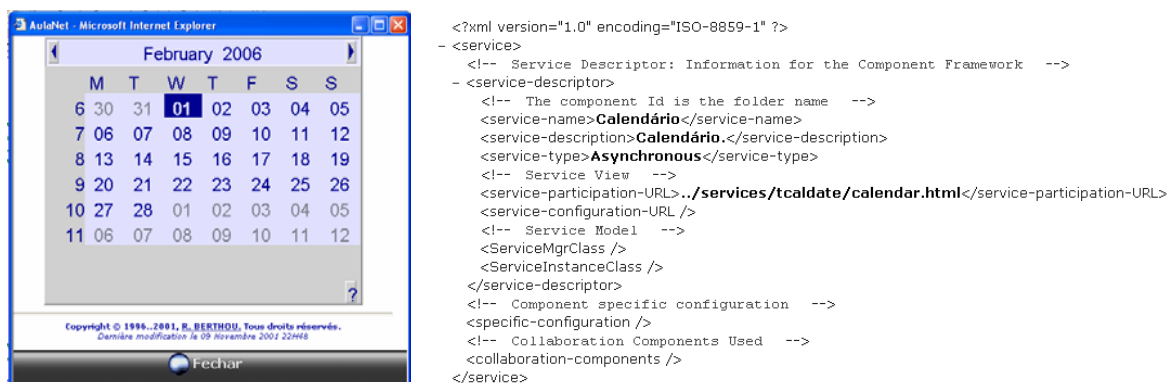
Figura 5.8. Adaptação de um serviço não desenvolvido originalmente para o AulaNet

Nas próximas duas subseções são ilustradas a componentização dos serviços Conferências e Debate ilustrando como a utilização de componentes 3C oferece suporte à montagem e à evolução dos serviços, e propicia o reuso. São enfocados os serviços de comunicação Conferências e Debate, que são os serviços para os quais foram direcionados mais tempo de pesquisa e de desenvolvimento no AulaNet. Mesmo sendo serviços de comunicação, possuem diversas funcionalidades de comunicação, coordenação e cooperação. Os diversos serviços compartilham funcionalidades dos três Cs.

5.1.4. Composição do serviço Conferências

O suporte à colaboração de um serviço evolui com o tempo. A Figura 5.9 apresenta a incorporação de novas funcionalidades ao longo do tempo no serviço Conferências. Na versão 2.0 do AulaNet, as funcionalidades foram desenvolvidas por diversos pesquisadores e grande parte delas foi replicada em outros serviços

[20] http://www.javaside.com/u_tcaldate.html

do ambiente, com pouco reuso do código. Neste serviço ao longo do tempo atuaram diretamente diversos alunos de mestrado e de doutorado, sendo que vários deles nem se conheceram[21]. Utilizar componentes para encapsular o suporte à colaboração possibilita o reuso de funcionalidades nos diversos serviços e uma maior independência no desenvolvimento.
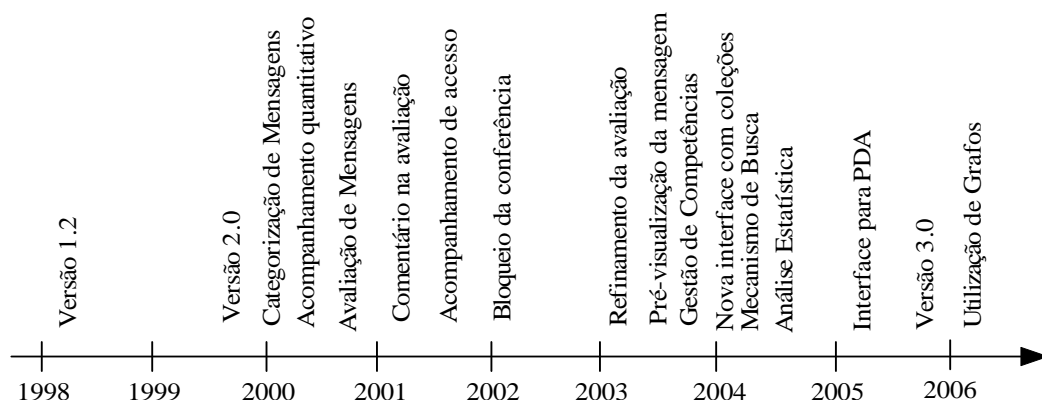
Figura 5.9. Novas funcionalidades incorporadas ao serviço Conferências ao longo do tempo

A Figura 5.10 apresenta novamente a dinâmica de uso do serviço Conferências no estudo dos conteúdos do curso TIAE. Conforme apresentado no Capítulo 3, o mediador seleciona o seminarista e declara a sessão iniciada; o seminarista submete o seminário e as questões, que são respondidas e discutidas pelos aprendizes; e o mediador avalia a mensagem e finaliza a sessão ao expirar o prazo. Para montar um serviço específico às necessidades da colaboração, selecionam-se os componentes 3C correspondentes.

---

[21] Atuo no desenvolvimento das Conferências desde o primeiro semestre de 2000, quando inicie meu mestrado sobre categorização de mensagens em ferramentas assíncronas [Gerosa, 2002].
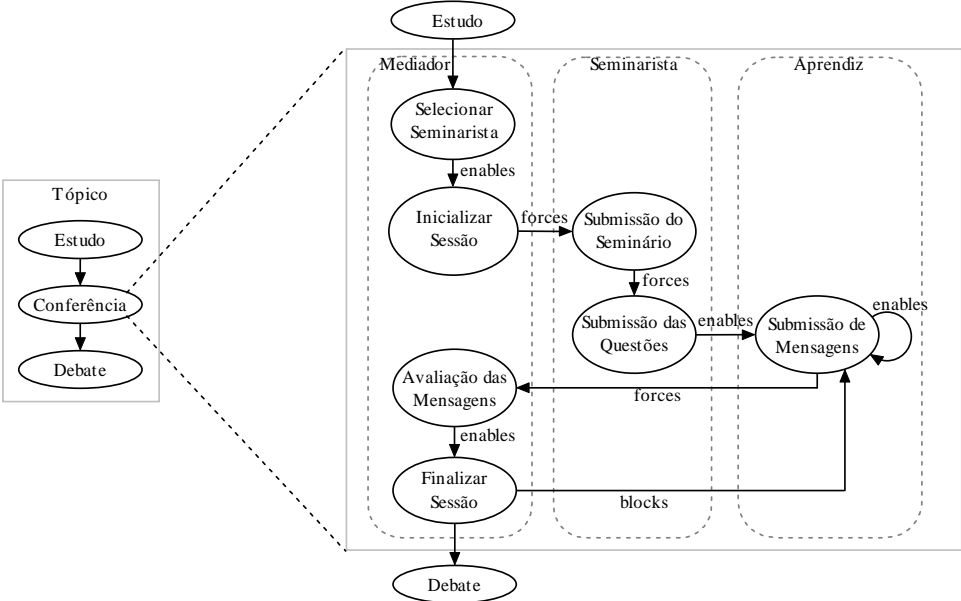
Figura 5.10. Dinâmica do uso do serviço Conferências no curso TIAE

Para implementar as funcionalidades requeridas para oferecer suporte computacional a estas atividades são utilizados os componentes 3C apresentados na Tabela 5.3. Os componentes implementam as funcionalidades existentes na versão atual das Conferências, mantendo a mesma interface com o usuário.

	Componente 3C	Propósito
Comunicação	MessageMgr	lidar com as mensagens do serviço
	CategorizationMgr	implementar a categorização de mensagens
	DialogStructureMgr	propiciar a estruturação de mensagens em árvores
Coordenação	SessionMgr	gerenciar as sessões de utilização do serviço, sendo uma sessão para cada tópico
	AssessmentMgr	cuidar da avaliação de mensagens
	RoleMgr	implementar os diversos papéis
	PermissionMgr	controlar as permissões de cada papel
	ParticipantMgr	lidar com os participantes do serviço
	CompetencyMgr	levar em consideração a participação no serviço no cálculo da performance do participante
Cooperação	CooperationObjMgr	cuidar da persistência e do sincronismo
	AccessRegistrationMgr	possibilitar que o participante seja informado do que já acessou
	StatisticalAnalysisMgr	implementar os recursos de análise estatística das mensagens
	SearchMgr	possibilitar a busca de mensagens

Tabela 5.3. Componentes 3C para dar suporte computacional à dinâmica das Conferências

Os componentes foram selecionados para atender às necessidades e características do serviço. As Conferências utilizam uma estruturação de mensagens em árvore, por favorecer a organização e a reflexão. A categorização de mensagens, no caso do TIAE, é utilizada para complementar a estruturação,

provendo uma melhor organização ao volume de mensagens. Os papéis utilizados nas Conferências do TIAE, seminarista e moderador, não possuem suporte computacional na versão 2.0 do ambiente e, portanto, não foram transpostos para a instanciação do serviço na nova versão. Foram mantidos apenas os papéis coordenador, mediador e aprendiz, já presentes na versão anterior. A avaliação de mensagens é utilizada nas Conferências para possibilitar a conceituação individual de cada mensagem e o registro de comentários visíveis para o autor, para a turma ou para os mediadores. O serviço Relatórios de Participação consolida o acompanhamento da participação a partir dos dados das avaliações dos diversos serviços. Os relatórios são visíveis para os aprendizes, que acompanham seu progresso e comparam-no com o de seus colegas.

Se houver alterações na dinâmica de colaboração, o serviço é recomposto ou reconfigurado, alterando o conjunto de componentes 3C ou customizando-os. Por exemplo, a dinâmica da colaboração no serviço pode ser alterada para incluir novas meta-informações a serem preenchidas ao escrever as mensagens; um sistema de workflow, para acompanhar o fluxo de tarefas realizadas no serviço; e um mecanismo de recomendação para que os aprendizes votem e recomendem mensagens. Estas alterações são mapeadas em componentes de comunicação, coordenação e cooperação, respectivamente.

Os componentes 3C são agregados ao serviço de modo a reusar o suporte computacional à colaboração. Grande parte das alterações apresentadas na Figura 5.9 são mapeadas a componentes 3C. Algumas funcionalidades, como pré-visualização, manipulação de coleções e interface para PDA, atingem somente a interface com o usuário, não sendo contempladas pelos componentes 3C, que são voltados para a camada de negócio da aplicação. A Tabela 5.4 apresenta o mapeamento das funcionalidades apresentadas na Figura 5.9 e os componentes 3C correspondentes.

Funcionalidade	Componente 3C
Categorização	CategorizationMgr
Acompanhamento quantitativo	AwarenessMgr
Avaliação de mensagens	AssessmentMgr
Comentário na avaliação	AssessmentMgr
Acompanhamento de acesso	AccessRegistrationMgr
Bloqueio da conferência	DiscreteChannelMgr
Refinamento da avaliação	AssessmentMgr
Pré-visualização	-
Gestão de competências	CompetencyMgr
Nova interface com coleções	-
Mecanismo de busca	SearchMgr
Análise estatística	StatisticalAnalysisMgr
Interface para PDA	-
Utilização de grafos	DialogStructureMgr

Tabela 5.4. Mapeamento das funcionalidades incorporadas ao serviço Conferências aos componentes 3C correspondentes

Os componentes plugados à arquitetura do serviço possibilitam flexibilizar e experimentar o suporte à colaboração. Serviços são construídos sob demanda, evitando incluir funcionalidades desnecessárias para o grupo em questão. A comunicação, a coordenação e a cooperação são abordadas e analisadas separadamente, de modo a definir o suporte à colaboração no grupo. Com a realimentação obtida a partir da utilização do serviço na colaboração, refina-se o conjunto de componentes adotado e sua configuração, adaptando o suporte computacional à evolução do grupo e da dinâmica adotada.

Eventualmente, uma alteração da dinâmica da colaboração leva à substituição de um componente 3C por uma versão mais robusta. Pode-se notar na evolução do serviço Conferências (Figura 5.9) que houve um determinado momento onde o suporte computacional à avaliação de mensagens foi refinado. Inicialmente, o conceito médio da participação de um aprendiz nas conferências era a média aritmética das avaliações de todas suas mensagens. Entretanto, este modelo não leva em consideração a quantidade de mensagens enviadas, de modo que se o aprendiz enviar uma única mensagem muita boa, sua avaliação final tem o mesmo conceito. Implementou-se então um balanceamento entre a quantidade de mensagens e a nota final correspondente [Pimentel et al., 2004]. Os modelos que foram disponibilizados, conforme ilustrado na Figura 5.11 são: Quantidade Irrelevante, onde a quantidade de mensagens não influencia o valor da nota final; Quantidade Mínima, onde a nota é reduzida progressivamente se não for enviada pelo menos uma certa quantidade de mensagens; Quantidade Máxima, onde a nota é reduzida para valores acima ou abaixo de uma determinada quantidade; e

Moderada, onde a nota é reduzida se a quantidade de mensagens for abaixo de um valor mínimo ou acima de um máximo. A adoção de um máximo inibe que participantes monopolizem a discussão.
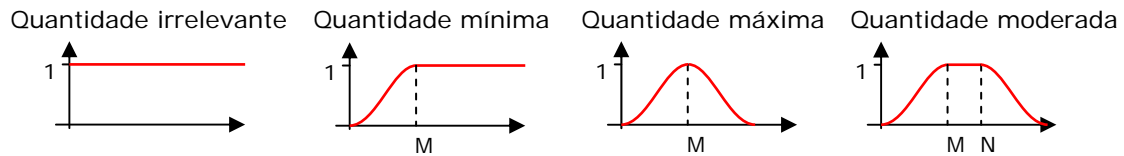
Figura 5.11. Fator de correção aplicado à nota final em função da quantidade de mensagens para os diferentes modelos de avaliação

Um novo componente pode ser desenvolvido e implantado sem que seja necessário alterar os serviços que utilizam o componente anterior. Desta forma, a utilização de componentes possibilita manter as duas versões do suporte à avaliação disponíveis, sendo utilizadas por serviços diferentes. Manter vários componentes com abordagens diferentes para o mesmo propósito dá mais flexibilidade ao desenvolvedor, que seleciona o que melhor atende às necessidades do serviço em questão. Possibilita também o refinamento de um dos aspectos da colaboração, substituindo o componente correspondente por outra versão com uma abordagem ou funcionalidade diferente. Entretanto, os componentes devem ser compatíveis entre si para reutilizar os dados.

Por delegar grande parte do suporte computacional à colaboração aos componentes 3C, que são reusados por diversos serviços, o código do serviço é reduzido. O serviço Conferências na versão 2.0 do AulaNet possui 4279 linhas de código exclusivas, não reusadas por outros serviços do ambiente. Na nova versão, o serviço possui 2905 linhas de código não reusadas, sendo que apenas 317 são referentes à camada de negócio, onde os componentes 3C promovem o reuso. As 2588 linhas restantes são referentes à interface com o usuário, que foi pouco alterada da versão 2.0 para a 3.0. Com a futura utilização de componentes de interface seguindo a especificação Java Server Faces, espera-se uma grande redução também nesta quantidade de código. A redução da quantidade de código a manter e a modularização do suporte computacional à colaboração facilitam a evolução futura do serviço.

5.1.5. Composição do Serviço Debate

O serviço Debate é utilizado no curso TIAE para consolidar o tópico semanal e para promover o alinhamento de idéias entre os aprendizes. A versão 1.0 do serviço, utilizado nas primeiras edições do curso, apresenta uma implementação típica para um chat, conforme ilustrado na Figura 5.12. Há uma caixa de texto para digitação da mensagem, uma área de texto onde as mensagens são postadas e uma lista dos participantes conectados. Este serviço é utilizado para uma discussão guiada pelo moderador da semana, auxiliado pelos mediadores.
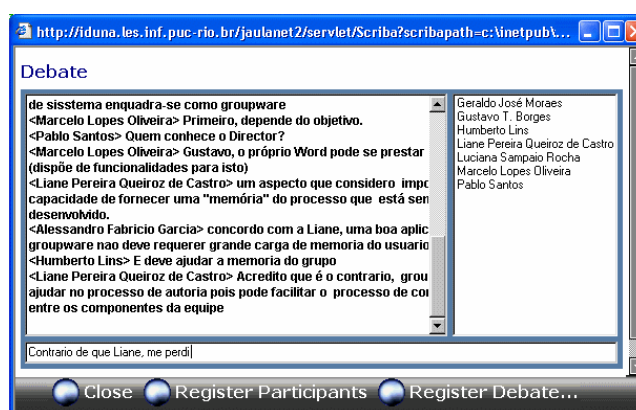
Figura 5.12. A versão 1.0 do serviço Debate do AulaNet

Até recentemente, não havia uma dinâmica pré-estabelecida para a discussão, ficando o moderador livre para conduzir o debate. Entretanto, para um melhor aproveitamento e organização da conversação e para reduzir a dependência do sucesso do Debate da habilidade do moderador, resolveu-se estabelecer uma dinâmica, conforme ilustrado na Figura 5.13 [Rezende et al., 2003]. Na nova dinâmica, o moderador do debate é incumbido de postar um resumo do que foi discutido ao longo da semana e iniciar um processo de apresentar questões e discuti-las, até que o mediador declara a sessão finalizada. Para cada questão discutida, os aprendizes postam um comentário, todos votam em qual contribuição será discutida, e então se inicia um período de discussão livre.
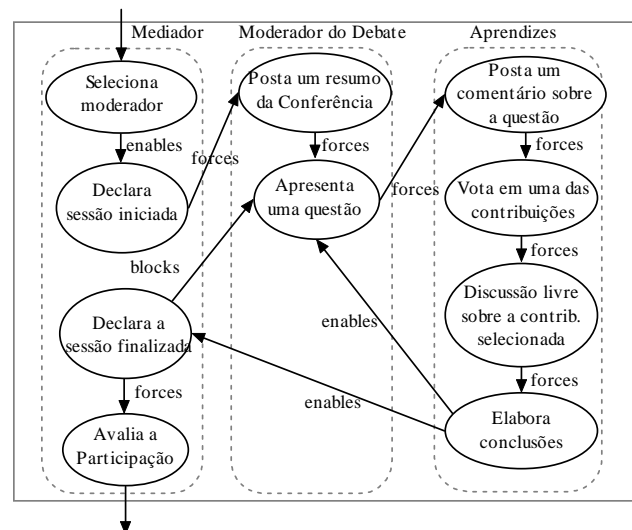
Figura 5.13. Nova dinâmica adotada no debate do curso TIAE

Ao executar esta dinâmica na versão 1.0 do Debate, foram encontrados diversos problemas [Pimentel et al., 2003a]: vários aprendizes mandavam mensagens juntamente com o moderador, mensagens chegavam após a questão ter sido considerada concluída, gastava-se tempo para organizar as propostas de discussão e a votação, havia uma sobrecarga de mensagens, etc. Estes problemas causavam confusão na conversação e a perda de co-texto [Pimentel et al., 2003b]. Resolveu-se incrementar o serviço para tratar estes problemas.

Foi feita uma análise da colaboração em função do modelo 3C, e decidiu-se em um primeiro momento investir no aprimoramento da coordenação do grupo. Para instrumentar o protocolo social vigente foram disponibilizadas novas informações de percepção para dar ao grupo uma melhor ciência do que está acontecendo e do que aconteceu anteriormente. A visualização do tema em discussão, o horário de envio da mensagem, a identificação dos mediadores e suas mensagens, a informação de quem está escrevendo e a ordem de participação foram incorporados à interface com usuário. A partir destes elementos, os participantes têm mais informações para decidir o momento de agir.

Para gerenciar o acesso ao canal de comunicação, foram disponibilizados mecanismos de coordenação, para serem utilizados para organizar a discussão. Para facilitar a intervenção do mediador, sem sobreposição de mensagens, e para evitar a continuação da discussão depois que o moderador mudou o assunto, implementou-se no Debate a possibilidade de o mediador travar o envio de mensagens pelos aprendizes e de estabelecer técnicas de conversação, que

definem o escalonamento da participação. As técnicas disponíveis são: Contribuição Livre, Contribuição Circular e Contribuição Única. A interface do serviço é ilustrada na Figura 5.14.
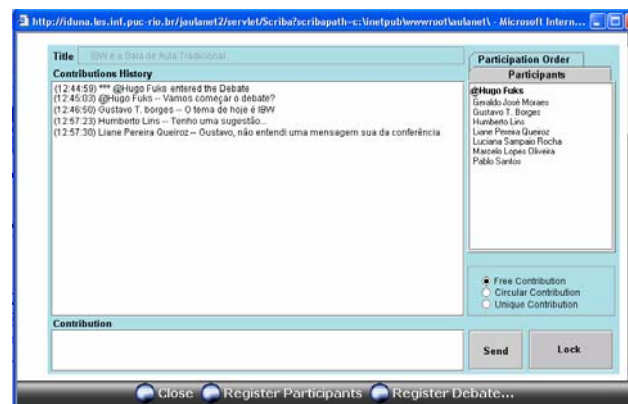
Figura 5.14. A versão 2.0 do Debate

Para executar a tarefa de postar um comentário sobre a questão é utilizada a técnica de Contribuição Circular, onde há uma ordem de participação e cada aprendiz envia uma mensagem na sua vez. Para votar em uma contribuição para ser discutida, é utilizada a técnica Contribuição Única, de modo que cada participante envia uma única contribuição, pois o canal fica inabilitado após o envio. Nas tarefas do mediador e do moderador do debate, o canal é bloqueado para os demais aprendizes. Nas demais tarefas é utilizada a técnica de Contribuição Livre.

A versão mais simples do Debate foi mantida para ser utilizada em cursos onde seja adotada uma dinâmica menos estruturada. A versão simplificada do Debate foi chamada de Bate-Papo. A Tabela 5.5 compara os componentes 3C utilizados no Bate-Papo e no Debate. O Debate 2.0 acrescenta o componente FloorControlMgr, que lida com as técnicas de conversação, e utiliza o componente AwarenessMgr, para gerenciar as informações de percepção. Este exemplo ilustra o aperfeiçoamento de um serviço de comunicação, sem alterar o suporte específico à comunicação, tratando apenas da coordenação. O desenvolvimento das versões do Debate e do Bate-Papo foi conduzido por Mariano Pimentel [2006] utilizando a abordagem proposta nesta tese.

Bate-Papo	Debate 2.0
MessageMgr	MessageMgr
DiscreteChannelMgr	DiscreteChannelMgr
AssessmentMgr	AssessmentMgr
RoleMgr	RoleMgr
PermissionMgr	PermissionMgr
ParticipantMgr	ParticipantMgr
SessionMgr	SessionMgr
CooperationObjMgr	AwarenessMgr
	CooperationObjMgr
	FloorControlMgr

Tabela 5.5. Componentes 3C utilizados no Bate-Papo e no Debate 2.0

Após a implementação do Debate 2.0, passou-se a investigar a influência de outros elementos da colaboração na discussão. Em versões subseqüentes, foram investigados outros problemas, como sobrecarga de mensagens, problemas na interface escrita-leitura, descontextualização e mensagem particular em local público [Pimentel et al., 2004], além da integração do serviço com a gestão de competências do ambiente. Foram acrescentados os componentes 3C: TextualMediaMgr e CompetencyMgr.

Este estudo de caso ilustrou a necessidade de estender a ferramenta colaborativa para acompanhar a evolução da dinâmica de trabalho. O uso do modelo 3C favoreceu a análise isolada das necessidades e deficiências de cada aspecto da colaboração e a composição de uma ferramenta mais aderente às características desejadas. As necessidades foram mapeadas em componentes que plugados à arquitetura do ambiente possibilitam flexibilizar e experimentar o suporte à colaboração.

A abordagem proposta favorece também o reuso da implementação do suporte à colaboração. O mesmo componente AssessmentMgr que é utilizado nas Conferências é utilizado também no Debate, porém com outros modelos e conceitos de avaliação. Da mesma maneira que na Conferência, utiliza-se um componente de avaliação mais simples ou mais robusto dependendo do modelo de avaliação adotado. Com esta abordagem, é possível também alterar a implementação dos componentes 3C de modo a alterar seu comportamento, simultaneamente nos diversos serviços que o utilizam, sem a necessidade de modificá-los. Para tanto, basta substituir o componente 3C correspondente.

A abordagem proposta, a arquitetura e os componentes estão sendo utilizados no desenvolvimento da nova versão do ambiente AulaNet. Alguns

vídeos ilustrando algumas das operações descritas nesta seção estão disponíveis no web site http://groupware.les.inf.puc-rio.br. Nas próximas seções são abordados alguns outros indícios da viabilidade da proposta e de seu entendimento e aplicabilidade por outras pessoas e em outras situações.

5.2. Aceitação Acadêmica

Comparar em termos de qualidade do produto resultante uma abordagem de desenvolvimento de groupware com outras, envolve a experimentação com diversos grupos de desenvolvedores utilizando as abordagens para desenvolver sistemas de porte considerável. Dada a grande quantidade de abordagens presentes na literatura e as restrições de recursos, nesta tese, optou-se por analisar a aceitação acadêmica da proposta em vez de comparar os produtos resultantes. Com esta finalidade, foram consideradas apresentações, revisões provenientes de conferências e revistas científicas, citações em trabalhos relacionados e a equipe de desenvolvedores da EduWeb, buscando indícios de aceitação e entendimento da proposta no meio acadêmico e empresarial.

A abordagem e a arquitetura propostas nesta tese foram apresentadas no Doctoral Colloquium do CRIWG (*International Workshop on Groupware*) de 2004 e 2005, coordenados pelos professores Hugo Fuks e Gert-Jan De Vreede respectivamente. Esta seção de “Aceitação Acadêmica” na tese foi sugerida pelo professor De Vreede durante o colloquium de 2005, como forma de complementar a avaliação da viabilidade da abordagem, dada a dificuldade de validar e comparar trabalhos na área de desenvolvimento de groupware. Sua sugestão foi de utilizar os pareceres dos revisores de conferências e revistas como indícios da viabilidade e originalidade do trabalho.

A abordagem e a arquitetura propostas nesta tese foram publicadas e apresentadas em conferências e revistas científicas nacionais e internacionais. A Tabela 5.6 apresenta os veículos onde foram publicados artigos diretamente relacionados à abordagem proposta nesta tese. Todos os veículos são auditados, com dois a quatro revisores por artigo. As referências completas destes artigos estão disponíveis em uma seção ao final das referências bibliográficas.

Veículo	**Título do Artigo**
WDBC 2001 - 1º Workshop de Desenvolvimento Baseado em Componentes	Um groupware baseado no ambiente AulaNet desenvolvido com componentes
JAI 2002 - XXI Jornada de Atualização em Informática	Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas
EBR 2002 - First Seminar on Advanced Research in Electronic Business	Engineering Groupware for E-Business
WDBC 2003 - 3º Workshop de Desenvolvimento Baseado em Componentes	Evoluindo para uma Arquitetura de Groupware Baseada em Componentes: o Estudo de Caso do Learningware AulaNet
Webmedia 2003 - Simpósio Brasileiro de Sistemas Multimídia e Web (trilha de trabalho cooperativo assistido por computador)	Do Modelo de Colaboração 3C à Engenharia de Groupware
Revista Informática na Educação: Teoria e Prática, Vol 7, No. 1	O Modelo de Colaboração 3C no Ambiente AulaNet
WDBC 2004 - 4º Workshop de Desenvolvimento Baseado em Componentes	O Uso de uma Arquitetura Baseada em Componentes para Incrementar um Serviço do Ambiente AulaNet
WCSCW 2004 - 1º Workshop Brasileiro de Tecnologias para Colaboração	Suporte à Coordenação e à Cooperação em uma Ferramenta de Comunicação Textual Assíncrona: Um Estudo de Caso no Ambiente AulaNet
SBIE 2004 - Simpósio Brasileiro de Informática na Educação	Uma Arquitetura para o Desenvolvimento de Ferramentas Colaborativas para o Ambiente de Aprendizagem AulaNet
CSCWiD 2005 - 9th International Conference on CSCW in Design	Towards an Engineering Approach for Groupware Development: Learning from the AulaNet LMS Development
WDBC 2005 - 5º Workshop de Desenvolvimento Baseado em Componentes	Componentes Baseados no Modelo 3C para o Desenvolvimento de Ferramentas Colaborativas
WCSCW 2005 - 2º Workshop Brasileira de Tecnologias para Colaboração	AulaNet 3.0: desenvolvendo aplicações colaborativas baseadas em componentes 3C
International Journal of Cooperative Information Systems (IJCIS), v.14, n.2-3	Applying the 3C Model to Groupware Development

Tabela 5.6. Veículos das publicações diretamente relacionadas a esta tese

Os veículos das publicações apresentados na Tabela 5.6 se concentram basicamente nas comunidades de CSCW (Webmedia 2003, WCSCW 2004, CSCWiD 2005, WCSCW 2005 e IJCIS), desenvolvimento baseado em componentes (WDBC 2001, 2003, 2004 e 2005) e informática na educação (Revista Informática na Educação e SBIE 2004), que são comunidades diretamente relacionadas a este trabalho. O trabalho também foi apresentado como um mini-curso do JAI 2002 e publicado no EBR 2002 (*Seminar on Advanced Research in Electronic Business*).

A Tabela 5.7 apresenta alguns trechos das declarações feitas pelos revisores dos artigos. Os demais trechos, que não foram apresentados, são referentes à estrutura do artigo ou à relevância para a conferencia em questão, não tendo

relação com a proposta em si[22]. Os comentários são apresentados em ordem cronológica. Todas as revisões são feitas de forma anônima. Alguns dos comentários são referentes a artigos não aceitos para publicação.

Veículo	**Comentário**
Middleware Conference 2003	“The paper presents an interesting case study of component based software development in the context of a commercial middleware platform (J2EE). The study is based on a groupware application that poses a number of requirements which can be suitably fulfilled with the use of component-based middleware as the development and runtime environment.” Revisor 3
JCSCW (2003)	“As described, the model is a substantial elaboration of a few remarks in a 1991 CACM article by Ellis et al. I can see that a lot of thought went into it, and I also credit the authors with a sophisticated understanding of the issues in this domain, they have read and digested the literature.” Revisor 1 “The 3C model seems to be an interesting contribution for the CSCW domain as a detailed refinement of the Clover model and Ellis's conceptual model of groupware” Revisor 2
CRIWG 2004	“The 3C collaboration model is a good synthesis of several similar approaches.” Revisor 2
CSCW 2004	“I also don't believe that simply applying the 3C model helps you in componentizing your software. There are certainly many other ways of structuring components which might be more practical from a componentization perspective.” Revisor 2
WCSCW 2004	“O artigo apresenta a nova arquitetura do aula net, baseada em componentes. Tal abordagem é bastante promissora, considerando pontos como reuso, flexibilidade, manutenção, "tailorability", qualidade, etc.”. Revisor 1
SBIE 2004	“O trabalho apresenta uma proposta de arquitetura para sistemas colaborativos que facilita a configuração de acordo com a necessidade de grupos específicos e possibilita a incorporação de novos componentes, o que é muito desejável dado a variedade de situações nas quais esses sistemas são usados.” Revisor 3
CSCWiD 2005	“The mentioned approach, having a component-base architecture and a 3C collaborative model is very well explained and seems very appealing.” Revisor 2
WDBC 2005	“O uso de componentes em aplicações colaborativas, particularmente o AulaNet(bastante conhecido) é uma excelente idéia para tornar mais flexível e facilitar sua evolução, principalmente considerando que nestes tipos de aplicações as mudanças ocorrem com mais freqüência e o consenso é mais difícil.” Revisor 1
SBIE 2005	“Does not provide any new ideas or any important contribution.” Revisor 2
IJCIS (2005)	“The paper basically discusses various aspects of the so-called 3C model, that is a model supporting the development of collaborative systems, an the application of such model to the development of some learningware for a web-based course.” Revisor 1

Tabela 5.7. Trechos das revisões de artigos relacionados a esta tese

Dos comentários listados na Tabela 5.7, apenas os revisores da CSCW Conference 2004 e do SBIE 2005 apresentaram críticas à abordagem. O primeiro acredita que há maneiras mais práticas de estruturar o software do ponto de vista da componentização. O segundo não vê nenhuma nova idéia ou contribuição na proposta. Por outro lado, os outros revisores vêem uma abordagem interessante e promissora. Um revisor da Middleware Conference 2003 acredita que os

[22] Os trechos completos das revisões foram disponibilizados em http://groupware.les.inf.puc-rio.br

requisitos do desenvolvimento do AulaNet realmente são propícios para o desenvolvimento baseado em componentes. Os revisores do JCSCW elogiam o refinamento feito do modelo 3C, que de acordo com eles leva em consideração a literatura da área, e vêem nele uma contribuição interessante. Um dos revisores do CRIWG 2004 acredita que o modelo 3C é uma boa síntese de várias abordagens similares. Um revisor do WCSCW 2004 acredita que a abordagem de desenvolvimento baseado em componentes fundamentados no modelo 3C de colaboração é promissora, levando em consideração fatores como reuso, flexibilidade, manutenção, “tailorability”, qualidade, etc. Um revisor do SBIE 2004 afirma que a arquitetura proposta propicia a adaptação para as necessidades de grupos específicos e possibilita a incorporação de novos componentes, o que é desejável dado a variedade de situações nas quais os sistemas colaborativos são utilizados. Um revisor do CSCWiD 2005 acredita que a proposta é bastante atrativa. Um revisor do WDBC 2005 afirma que o uso de componentes em aplicações colaborativas, particularmente no AulaNet, é uma excelente idéia para tornar o sistema mais flexível e facilitar sua evolução, o que é freqüentemente necessário. Um revisor do IJCIS, ao recomendar a aceitação do artigo, afirma que é feita uma discussão do modelo 3C, que serve de suporte para o desenvolvimento de sistemas colaborativos, em especial para learningware.

Outro indício da aceitação da proposta é relativo à quantidade de citações em artigos de pesquisadores externos ao grupo de pesquisa no qual esta tese está inserida. Apesar de este indício necessitar de alguns anos para se tornar mais preciso, foram encontrados 26 citações a artigos apresentados na Tabela 5.6. As citações foram encontradas em buscas em mecanismos da Internet. As referências estão listadas em ordem alfabética na Tabela 5.8.

Artigo	Citação
ASSUNÇÃO, C.F.F., REAMI, E.R., ZUFFO, M.K. & LOPES, R.D. (2004) "MediColl Ambiente Cooperativo para Apoio ao Diagnóstico Médico à Distância" *IX CBIS - Congresso Brasileiro de Informática na Saúde*, 2004, Ribeirão Preto.	JAI 2002
BRENNAND, C.A.R.L., LINS NETO, J.H. & FERNANDES, S.M.M. (2005) "ATCA - Ambiente de Trabalho Colaborativo Avançado", *Congresso Internacional de Qualidade em EAD: desafios para a transformação social*, São Leopoldo-RS, 01-03 de junho de 2005.	JAI 2002
CINELLI, G.B., ZAINA, L.A.M., BAPTISTA, C.M., SILVEIRA, R.M., RUGGIERO, W.V. & BRESSAN, G. (2005) "Acompanhamento de sessões de fórum através da ferramenta Faucon", *WCSCW 2005 Workshop Brasileiro de Tecnologias para Colaboração*, 2005, Juiz de Fora.	JAI 2002
CORDENONSI, A.Z., MULLER, F.M. & BASTOS, F.P. (2005) "O Ensino de Heurísticas e Metaheurísticas na área de Pesquisa Operacional sob a ótica da Educação Dialógica Problematizadora", *RENOTE - Revista Novas Tecnologia na Educação*, V. 3 Nº 1, ISSN 1679-1916.	JAI 2002
CRUZ NETO, G.G., GOMES, A.S. & TEDESCO, P.C.A.R. (2003) "Aliando Teoria da Atividade e TROPOS na Elicitação de Requisitos" *Workshop de Engenharia De Requisitos*, 2003, Piracicaba, v. 1., pp. 63-77.	JAI 2002
CRUZ NETO, G.G., GOMES, A.S. & TEDESCO, P.C.A.R. (2003) "Elicitação de Requisitos de Sistemas Colaborativos de Aprendizagem Centrada na Atividade de Grupo", *Simpósio Brasileiro de Informática na Educação*, 2003, Rio de Janeiro.	JAI 2002
GIRARDI, R.A.D. (2004) Framework para coordenação e mediação de Web Services modelados como Learning Objects para ambientes de aprendizado na Web, *Dissertação de Mestrado*, Departamento de Informática – PUC-Rio.	WDBC 2003
GONÇALVES, B.S. & PEREIRA, A.C. (2004) "Color learning based on VLE-AD platform", *Proceedings of the AIC 2004 Color and Paints*, Interim Meeting of the International Color Association, pp 311-314.	JAI 2002
GONÇALVES, B.S. (2004) Cor Aplicada ao Design Gráfico: Um Modelo de Núcleo Virtual para Aprendizagem Baseado na Resolução de Problemas, *Tese de doutorado em Engenharia de Produção*, Universidade Federal de Santa Catarina.	JAI 2002
HEIDRICH, F.E. (2004) O uso do Ciberespaço na Visualização da Forma Arquitetônica de Espaços Internos em Fase de Projeto, *Dissertação de mestrado*, Programa de Pós-Graduação em Arquitetura e Urbanismo, Universidade Federal de Santa Catarina.	JAI 2002
JATOBÁ, P.H.G., LIMA, R., VILAR, G., OLIVEIRA, E.A. & MATTOS, S. "Collaborative Environments for Telecardiology", *25th Annual International Conference for Medicine and Biology Society*, IEEE, Cancun, pp. 291.	JAI 2002
JATOBÁ, P.H.G., LIMA, R.C.A., VILAR, G., MADEIRO, F., CAVALCANTI JR., A.L.O. (2005) "Modelo de um sistema colaborativo de telecardiologia", *Colabor@*, Vol 3, no 10, Novembro de 2005, ISSN 1519-8529	JAI 2002
LIMA, P.S.R., BRITO, S.R., SILVA, O.F. & FAVERO, E.L. (2005) "Adaptação de Interfaces em Ambientes Virtuais de Aprendizagem com Foco na Construção Dinâmica de Comunidades", *Novas Tecnologias na Educaçã*o, CINTED-UFRGS, Vol 3, No 1, Maio 2005.	WDBC 2003
MARCZAK, S.S. & GIRAFFA, L.M.M. (2003) "A Gerência da Informação em Ambientes de Ensino a Distância: um estudo comparativo", *Relatório Técnico*, N. 29, Porto Alegre: PPGCC FACIN PUCRS, Agosto 2003.	JAI 2002
MENEZES, C. S., MESQUITA, L.F. , PESSOA, J.M. & VESCOVI-NETTO, H. (2003) "Percepção em Comunidades Virtuais: Mantendo-se Antenado no AmCorA" *XIV Simpósio Brasileiro de Informática na Educação*, 2003, Rio de Janeiro, pp. 265-274.	JAI 2002
MIRANDA, I.S., ARAUJO, R.M. & SANTORO, F.M. (2005) "An Approach for Defining System Requirements for Group", *Anais do WCSCW - Workshop Brasileiro de Tecnologias para Colaboração*, 2005.	IJCIS
MOURA, J.G. & BRANDÃO, L.O. (2005) "Aplicações no SAW - Sistema de Aprendizagem pela Web", *Simpósio Brasileiro de Informática na Educação - SBIE 2005*, Juiz de Fora.	RIE 2004
MOURA, J.G., BRANDÃO, L.O. (2005) "SAW - Sistema de aprendizagem pela web: incorporando novos recursos", *WebMedia 2005 - Simpósio Brasileiro de Sistemas Multimídia e Web*.	RIE 2004
OEIRAS, J.Y.Y. & ROCHA, H.V (2005) "Requirements for Computational Environments to Support Institutional Cooperative Work", *Anais do II Workshop TIDIA 2005*, São Paulo.	IJCIS
PELLEGRINI, G.F. (2004) Uma Abordagem Metodológica para Engenharia de Groupware Aplicada a Área da Saúde, *Tese de doutorado em Engenharia de Produção*, Universidade Federal de Santa Catarina.	JAI 2002
PEREIRA, A.T.C., GONÇALVES, B., SUZUKI, F.G. & ALVES, T. (2004) "Aprendizagem da cor baseada na plataforma AVA-AD" *CONAHPA- Congresso Nacional de Ambientes Hipermídia para Aprendizagem*, 2004, Florianópolis.	JAI 2002
POZZER, C. T., RAPOSO, A.B., SANTOS, I.H.F., CAMPOS, J.L.E. & REIS, L P. (2003) "CSVTool - A Tool for Video-Based Collaboration", *IX Simpósio Brasileiro de Sistemas Multimídia e Web - WebMidia 2003,* Salvador-BA, pp.353-367.	EBR 2002
SILVA, E.Q & MOREIRA, D.A. (2004) "Um Framework de Componentes para o Desenvolvimento de Aplicações Web Robustas de Apoio à Educação", *Simpósio Brasileiro de Informática na Educação 2004*, Manaus-AM, pp. 158-167.	WDBC 2003
SILVA, J.C.T. (2004) Um Modelo para Avaliação de Aprendizagem no Uso de Ferramentas Síncronas em Ensino Mediado pela WEB, *Tese de doutorado*, Departamento de Informática – PUC-Rio.	WDBC 2003
TEIXEIRA, B. & CHAGAS, E.F. (2005) "Co-Autoria: Avaliação e Proposta de Requisitos para Ferramentas Segundo o Modelo 3C", *Workshop de Informática na Escola*, Congresso da Sociedade Brasileira de Computação 2005.	JAI 2002
VILAR, G., OLIVEIRA, E.A., JATOBÁ, P.H.G. & VILAR, D.S. (2004) "Processos colaborativos e tecnologias da informação aplicados ao ensino de medicina", *Colabor@*, n. 7, v. 2, 2004	JAI 2002

Tabela 5.8. Citações a artigos diretamente relacionados a esta tese

A abordagem proposta nesta tese, a arquitetura e os componentes também foram apresentados à equipe técnica responsável pelo AulaNet da empresa EduWeb. Estavam presentes na apresentação o diretor de tecnologia, sr. Luidi Xavier Fortunato, e o gerente de desenvolvimento do produto, sr. Reubem Alexandre Almeida Girardi. Ambos aprovaram a proposta, sem restrições. Eles ressaltaram que além de atender os requisitos para a nova versão do ambiente, a abordagem propiciará um menor custo de manutenção e possibilitará um melhor gerenciamento das diferentes versões com customizações e adaptações que eles mantêm para seus clientes.

Os resultados obtidos indicam a aceitação acadêmica da abordagem proposta nesta tese. Em geral, os pareceres dos especialistas foram positivos e houve aceitação da abordagem em veículos nacionais e internacionais, bem como uma quantidade razoável de citações em artigos publicados por outros grupos de pesquisa.

5.3. Estudo de Caso na Disciplina Engenharia de Groupware

Para obter mais indícios da aplicabilidade da proposta, ela foi utilizada na disciplina Engenharia de Groupware, cujos códigos são INF 1637 e INF 2132 para a graduação e pós-graduação do Departamento de Informática da PUC-Rio respectivamente. Leciono a disciplina juntamente com os professores Hugo Fuks e Alberto Barbosa Raposo desde o primeiro semestre de 2002. O estudo de caso foi realizado na turma do segundo semestre de 2005, com 2 alunos de graduação, 3 de mestrado e 2 de doutorado. O resultado do estudo de caso foi avaliado pela observação direta dos professores do curso e pela aplicação de questionários individuais.

Conforme apresentado no enunciado da Figura 5.15, o trabalho aplicado na disciplina é dividido em três partes. Na primeira parte, após o estudo dos elementos do modelo 3C, cada aluno seleciona uma aplicação e analisa suas funcionalidades, classificando-as em comunicação, coordenação ou cooperação. O aluno apresenta as telas da aplicação e justifica suas escolhas. A segunda parte do trabalho consiste em efetuar um estudo comparativo de aplicações colaborativas

similares à escolhida, fornecendo subsídios para o aprimoramento da ferramenta. Para este aprimoramento, o aluno propõe a incorporação de serviços e funcionalidades que estendam e complementem os três Cs analisados anteriormente. Na terceira parte do trabalho, o aluno apresenta uma arquitetura e um protótipo que ofereça suporte à extensão do sistema, utilizando a infra-estrutura e os componentes 3C propostos nesta tese.

Enunciado dos Trabalhos de Engenharia de Groupware

10/08/2005

Trabalho 1: Análise de uma aplicação com base no modelo 3C (05/10/2005)

O aprendiz deverá analisar os serviços oferecidos por uma aplicação não necessariamente colaborativa, cuja escolha será negociada com os mediadores. Deverá classificar os serviços desta aplicação em comunicação, coordenação ou cooperação. Também deverão ser analisados elementos de cada um dos Cs.

O aprendiz deverá apresentar o trabalho em sala de aula no dia marcado. O aprendiz deverá deixar bem claro nos slides de sua apresentação a classificação dos serviços e elementos em função de cada C, bem como apresentar telas da aplicação que justifique esta classificação.

Trabalho 2: Incorporação de serviços à aplicação (26/10/2005)

O aprendiz deverá propor a incorporação de serviços que aprimorem e complementem os 3C's à aplicação do trabalho anterior.

A primeira etapa do Trabalho 2 consiste em fazer um estudo comparativo de aplicações colaborativas similares à escolhida, fornecendo os subsídios para o aprimoramento solicitado acima. O aprendiz deverá apresentar o trabalho em sala de aula no dia marcado.

Trabalho 3: Projeto baseado no modelo 3C (07/12/2005)

O aprendiz deverá apresentar uma arquitetura e um protótipo que dê suporte ao sistema proposto no trabalho 2. O aprendiz deverá apresentar o trabalho em sala de aula no dia marcado e entregar uma documentação contemplando o conteúdo dos três trabalhos.

Figura 5.15. Enunciado dos trabalhos da disciplina Engenharia de Groupware

Na primeira etapa do trabalho, houve alguns problemas de classificações decorrentes de entendimento incorreto do modelo. Praticamente em todos trabalhos ocorreram classificações incorretas, que foram apontadas pelos professores do curso e corrigidas pelos alunos. A Tabela 5.9 apresenta a quantidade de funcionalidades identificadas correta e incorretamente em cada trabalho. Pode-se notar uma porcentagem de acerto médio de 76%. Este resultado foi considerado satisfatório, visto que não foram apresentados e discutidos com os alunos exemplos e critérios de classificação antes da apresentação do trabalho. A dinâmica do curso prevê que neste primeiro trabalho surjam dificuldades e problemas para serem explorados em discussões subseqüentes, visando a consolidação do aprendizado. Na segunda etapa do trabalho, a maior parte dos

alunos conseguiu classificar corretamente as novas funcionalidades que estavam sendo propostas, indicando o entendimento do modelo.

Ferramenta	Funcionalidades classificadas	Funcionalidades incorretamente classificadas
Stratify	6	2
GoogleTalk	6	3
BuddySpace	16	4
Orkut	5	1
Senhor da Guerra	3	1
MSN	14	1
TortoiseSVN	9	2

Tabela 5.9. Identificação e classificação das funcionalidades das ferramentas escolhidas

Na terceira etapa, o suporte computacional às necessidades identificadas deveria ser projetado levando em consideração o *Collaboration Component Kit* proposto nesta tese, cuja documentação foi entregue aos alunos. Os alunos não tiveram que implementar a arquitetura proposta, pois não haveria tempo hábil no escopo da disciplina.

O aluno que analisou a ferramenta Stratify[23], que é voltada à construção de taxionomias, propôs a incorporação de uma nova funcionalidade de comunicação, duas de coordenação e uma de cooperação: um chat para discussão síncrona, cores para representar os participantes e os artefatos sob sua responsabilidade, indicação de quem está conectado e histórico de ações. Para isto, montou uma arquitetura utilizando os componentes DiscreteChannelMgr, MessageMgr, ParticipantMgr, GroupMgr, AwarenessMgr, SessionMgr e CooperationObjMgr. O componente AvailabilityMgr, específico para a gestão da disponibilidade dos participantes, não foi utilizado pelo aluno.

O aluno que analisou a ferramenta GoogleTalk[24], que é voltada para conversação por voz através da Internet, propôs a incorporação de três novas funcionalidades de comunicação e uma de cooperação: vídeo-chat, audioconferência para várias pessoas, envio de cartões virtuais e transferência de arquivos. Para isto, montou uma arquitetura utilizando os componentes VideoMediaMgr e PictorialMediaMgr e propôs um novo componente de cooperação: FileTransferMgr. Entretanto, não utilizou em sua arquitetura alguns

[23] http://www.stratify.com
[24] http://www.google.com/talk

componentes 3C necessários, como ParticipantMgr, SessionMgr e ContinousChannelMgr.

O aluno que analisou a ferramenta BuddySpace[25], que é voltada para comunicação pela Internet integrada à noção de contexto e localização, propôs duas novas funcionalidades de comunicação e três de coordenação: formatação do texto das mensagens, conversa por áudio, ordenação das pessoas que desejam falar, notificação de deslocamento e gestão de confiança e reputação. Para isto, montou uma arquitetura onde utilizou uma versão estendida do MessageMgr e os componentes AudioMediaMgr, MetaInformationMgr e ParticipantMgr. Além destes componentes, propôs novos componentes: WirelessMocaMgr, OrderMgr e ReputationMgr, sendo o primeiro para prover a integração com o framework MoCA [Sacramento et al., 2004], o segundo para a gestão da ordem de participação e o terceiro para a gestão da reputação no grupo. O OrderMgr foi proposto desnecessariamente, pois sua funcionalidade já é provida pelo componente FloorControlMgr.

O aluno que analisou o site Orkut[26], que é voltado para estabelecimento de redes de contatos e relacionamentos, propôs duas novas funcionalidades de comunicação e uma de coordenação: um chat, uma audioconferência e uma lista de participantes conectados. Para isto, montou uma arquitetura onde utilizou os componentes MessageMgr, DiscreteChannelMgr e AudioMediaMgr. Propôs um novo componente: OnlineMgr, para gerenciamento dos participantes presentes. Entretanto, esta funcionalidade já é provida pelo componente AwarenessMgr em conjunto com o SessionMgr.

O aluno que analisou o jogo O Senhor da Guerra[27], que é jogado através de mensagens SMS, propôs a criação de um novo jogo com uma funcionalidade de comunicação, cinco de coordenação e duas de cooperação: envio de mensagens para os jogadores, ranqueamento, agrupamento em clãs, gerenciamento de tarefas, posicionamento das naves, definição de objetivos, mapa do universo e batalha conjunta. Para isto, propôs seus próprios componentes, mesmo havendo componentes 3C que atenderiam suas necessidades, como MessageMgr,

[25] http://buddyspace.sourceforge.net
[26] http://www.orkut.com
[27] http://www.senhordaguerra.com.br

DiscreteChannelMgr, RoleMgr, PermissionMgr, ParticipantMgr, GroupMgr, SessionMgr, FloorControlMgr, TaskMgr, AwarenessMgr, CooperationObjMgr e RankingMgr.

O aluno que analisou a ferramenta MSN[28], que oferece serviços de comunicação síncrona, propôs duas novas funcionalidades de coordenação e duas de cooperação: utilização de níveis de disponibilidade para diferentes grupos de contatos, possibilidade de bloquear o envio de mensagens enquanto um dos participantes estiver escrevendo, registro do horário de envio das mensagens e possibilidade de envio de arquivo para várias pessoas simultaneamente. Para isto, montou uma arquitetura onde utilizou os componentes DiscreteChannelMgr, GroupMgr, FloorControlMgr, AwarenessMgr, MessageMgr e uma versão estendida do CooperationObjMgr, para possibilitar a transferência de arquivos, e do AvailabilityMgr, para configurar as diferentes disponibilidades.

O aluno que analisou a ferramenta TortoiseSVN[29], que possibilita integrar o controle de versões aos arquivos do sistema operacional, propôs uma nova funcionalidade de comunicação e uma de coordenação: discussão assíncrona sobre as mudanças efetuadas e notificação sobre atualização de artefatos. Para isto, montou uma arquitetura utilizando os componentes MessageMgr e DialogStructureMgr, e propondo os componentes MessageStoreMgr e RevisionMgr. As funcionalidades destes componentes são oferecidas pelos componentes CooperationObjMgr e AwarenessMgr, não sendo necessária sua criação.

Em geral, os alunos utilizaram corretamente a abordagem, analisando a ferramenta escolhida e algumas ferramentas similares em função do modelo 3C, para sugerir a incorporação de novas funcionalidades visando aprimorar o suporte computacional à colaboração. Entretanto, alguns alunos tiveram dificuldades na seleção dos componentes e na montagem da arquitetura. Conforme ilustrado na Tabela 5.10, desconsiderando o aluno que projetou o jogo, que entendeu incorretamente o enunciado do trabalho, 76% dos componentes foram selecionados e utilizados adequadamente. Separando alunos do mestrado e

[28] http://www.microsoft.com/msn
[29] http://tortoisesvn.sourceforge.net

doutorado dos alunos da graduação, esta porcentagem fica em 85% para os primeiros e em 50% para os últimos. Dada a quantidade reduzida de alunos participando no experimento, não é possível obter resultados conclusivos. Entretanto, há indícios de que há necessidade de aprimorar a documentação dos componentes e de capacitar os alunos na utilização de componentes de software, visto que o assunto normalmente não é coberto nos cursos de graduação. Para os próximos semestres, a documentação será estendida e revista e a dinâmica do curso será modificada para que os alunos implementem um protótipo da solução, visto que grande parte dos erros seria perceptível, caso eles tivessem implementado a extensão proposta à ferramenta colaborativa. Apesar da ocorrência de problemas na resolução dos trabalhos, o experimento foi considerado satisfatório, pois os alunos entenderem e utilizaram a abordagem proposta nesta tese, mesmo não sendo da área.

Ferramenta	Novas Funcionalidades	Quantidade de Componentes	Componentes Utilizados Incorretamente
Stratify	4	8	1
GoogleTalk	4	6	3
BuddySpace	5	7	1
Orkut	3	5	2
Senhor da Guerra	8	-	-
MSN	4	7	0
TortoiseSVN	2	4	2

Tabela 5.10. Utilização dos componentes 3C

Ao final da apresentação da terceira etapa do trabalho, os alunos receberam e responderam um questionário sobre o ferramental disponibilizado. O resultado do questionário é apresentado na Tabela 5.11. A maior parte dos alunos avaliou o grau de dificuldade da utilização do modelo 3C para a análise da aplicação escolhida como regular, em uma escala de muito difícil a muito fácil. O entendimento do modelo 3C teve a mesma avaliação. Estes resultados foram considerados satisfatórios, visto que os alunos tiveram o primeiro contato com o modelo 3C e com a abordagem durante o curso e não são especialistas em groupware. Com relação à abrangência do modelo 3C na modelagem do sistema, 5 alunos avaliaram como suficiente e 2 como regular, indicando que a maior parte das funcionalidades identificadas foram classificadas. Com relação à utilização dos componentes 3Cs, 5 alunos identificaram a solução como complexa e 2 como normal, em uma escala de muito simples a muito complexa. Este resultado também foi considerado satisfatório, dado que, além de não serem especialistas

em groupware, os alunos não são especialistas em componentes de software. Apesar de a maioria ter classificado a solução como complexa, todos avaliaram a utilização de componentes 3C na composição de groupware como bom ou muito bom. Com relação ao encapsulamento das complexidades de baixo nível, 3 alunos avaliaram como neutro, 2 como bom e 2 como muito bom. Por fim, ao avaliar o suporte computacional a groupware utilizando componentes 3C, 2 alunos avaliaram como neutro e 5 como bom. Os resultados obtidos no questionário foram em geral positivos.

Pergunta	Resultado		
A análise da aplicação escolhida para o trabalho através do modelo 3C foi:	(0) Muito difícil (1) Fácil	(0) Difícil (0) Muito fácil	(6) Regular
O entendimento do modelo 3C é:	(0) Muito difícil (1) Fácil	(0) Difícil (0) Muito fácil	(6) Regular
A abrangência do modelo 3C na modelagem do sistema é:	(0) Muito insuf. (5) Suficiente	(0) Insuficiente (0) Muito suficiente	(2) Regular
Como você classifica a solução com componentes baseados no modelo 3C?	(0) Muito limitada (5) Complexa	(0) Limitada (0) Muito complexa	(2) Normal
Componentes 3C na composição de groupware é:	(0) Muito ruim (5) Bom	(0) Ruim (2) Muito bom	(0) Neutro
O encapsulamento das complexidades de baixo nível de sistemas multi-usuário é	(0) Muito ruim (2) Bom	(0) Ruim (2) Muito bom	(3) Neutro
O suporte computacional a groupware usando os componentes 3C é:	(0) Muito ruim (5) Bom	(0) Ruim (0) Muito bom	(2) Neutro

Tabela 5.11. Resultados dos questionários aplicados aos aprendizes

Além de ser utilizada pelos alunos da disciplina Engenharia de Groupware, a abordagem foi utilizada por outros membros da equipe do AulaNet do LES/PUC-Rio. Mariano Pimentel a utilizou no desenvolvimento da nova versão do serviço Debate, e Denise Filippo na construção da versão para PDA do AulaNet.

5.4.
Estudo de Caso com o TelEduc

O suporte computacional à colaboração do fórum de discussão do ambiente TelEduc pode ser mapeado aos componentes 3C. Na Tabela 5.12, as funcionalidades presentes na ferramenta e os componentes 3C correpondentes são apresentados.

Funcionalidade	Componente 3C
Mensagens	MessageMgr
Canal de Comunicação	DiscreteChannelMgr
Estruturação em árvore	DialogStructureMgr
Lixeira	TrashBinMgr
Avaliação	AssessmentMgr
Busca	SearchMgr
Permissões de acesso	PermissionMgr
Papéis	RoleMgr
Participantes	ParticipantMgr
Eventos	AwarenessMgr

Tabela 5.12. Mapeamento das funcionalidades do fórum de discussão do TelEduc

Mesmo possuindo interfaces com usuário complemente distintas, a camada de negócio da aplicação pode ser reusada de modo a compartilhar funcionalidades e o suporte à colaboração. Os componentes do Collaboration Component Kit podem ser utilizados para instanciar ferramentas de outros groupwares.

5.5.
Considerações Finais

Sistemas colaborativos são especialmente vulneráveis a falhas [Grudin, 1989], envolvem aspectos multidisciplinares em sua construção e são difíceis de desenvolver, manter, aplicar e testar. Isto é especialmente recorrente em ferramentas colaborativas para o ensino-aprendizagem, como o AulaNet, que normalmente envolve uma turma heterogênea, de curta duração e muitas vezes despreparada para a utilização da tecnologia no ensino-aprendizagem. A utilização de técnicas de desenvolvimento baseado em componentes é uma forma de desenvolver groupware extensível.

Através da componentização, o AulaNet se torna mais adaptável à variedade de grupos que o utilizam, que neste caso são compostos por alunos e professores de diferentes regiões do país e do mundo, e às diversas características dos cursos aplicados através dele, que variam no conteúdo (desde cursos para crianças até cursos de pós-graduação) e na abordagem pedagógica utilizada. Além disto, a dinâmica e o grupo evoluem com o tempo e não há como prever todas as demandas para o ambiente. A tecnologia de componentes é propícia para utilização em sistemas com requisitos instáveis ou imprecisos [Szyperski, 1997].

A nova versão do AulaNet provê um conjunto padrão de serviços adaptável para cada servidor. O desenvolvedor do ambiente utiliza componentes

interoperáveis para compor os serviços. São acoplados e desacoplados componentes de modo a montar ferramentas visando oferecer suporte às necessidades de colaboração de cada grupo. São experimentadas diversas configurações e, a partir da realimentação obtida com a utilização da ferramenta, é feita uma adaptação e refinamento, de modo a acompanhar a evolução do grupo e da dinâmica do curso.

O desenvolvimento baseado em componentes facilita a equipes externas ao projeto AulaNet, como pesquisadores de outras instituições, desenvolverem e adquirem novos serviços e incorporá-los ao ambiente. Para isto, não é necessário ter acesso ou entender o código fonte do ambiente, basta respeitar as interfaces e protocolos de comunicação definidos. Uma instituição pode desenvolver novos serviços e incorporá-los ao AulaNet. Passa a ser possível também encapsular ferramentas que não foram originalmente desenvolvidas para o ambiente, implementando um adaptador que faça a intermediação da comunicação e construindo um empacotamento de acordo com os padrões definidos no modelo de componentes. Além disto, pesquisadores podem usar a arquitetura componentizada do AulaNet para disponibilizar e testar seus protótipos de ferramentas colaborativas sem ter que construir um ambiente inteiro para isto. Ao utilizar um ambiente robusto, gratuito e largamente difundido, aumenta a chance de utilização e a abrangência da ferramenta desenvolvida.

Com equipes de outras instituições e empresas desenvolvendo novos serviços para o ambiente, a equipe AulaNet focará seus esforços na arquitetura, na integração dos diversos componentes e na pesquisa de novos mecanismos de colaboração. Instituições que forem usar o AulaNet o adequarão e o incrementarão de acordo com suas necessidades. Além disto, equipes especializadas poderão atuar nos diversos componentes e serviços do ambiente, o que vai ao encontro das necessidades da equipe de desenvolvimento do AulaNet na PUC-Rio, que é composta de alunos desenvolvendo seus trabalhos de curso. Normalmente, cada aluno está focado em um determinado aspecto da colaboração.

Na Seção 2.3 desta tese, foram apresentadas algumas dificuldades associadas ao uso da tecnologia de componentes freqüentemente citadas na literatura. Uma delas é relacionada com o esforço inicial de projeto e

implementação para montar a infra-estrutura do sistema e construir uma biblioteca robusta de componentes reusáveis. No caso desta tese, o custo inicial será amortizado nos sucessivos reusos e na esperada redução do custo de manutenção e evolução. As dificuldades relacionadas à incorporação de componentes provenientes de terceiros tem um impacto reduzido nesta tese, pois a maior parte dos componentes são desenvolvidos e utilizados internamente. Por fim, a dificuldade associada à mudança no processo de desenvolvimento está sendo tratada neste consórcio de pesquisa na tese de Mariano Pimentel [2006].

Na Seção 2.1.4 foi apresentada a arquitetura genérica para groupware proposta por Dewan [1998]. Na arquitetura de Dewan, o tratamento da semântica da aplicação é localizado em um servidor central, e em uma das camadas a arquitetura é ramificada para as estações clientes. Na arquitetura utilizada nesta tese, o servidor trata tanto a lógica de negócio quanto a construção da apresentação, pois os participantes interagem com o groupware através de um navegador web. A arquitetura apresenta um processamento da lógica de negócio nas estações clientes apenas nos casos onde o lado cliente do serviço é implementado através de um applet, como em algumas ferramentas de comunicação síncrona. Entretanto, por uma reivindicação da EduWeb, embasada em demandas do meio corporativo, os applets estão sendo evitados na nova versão do AulaNet, de modo a reduzir os problemas de configuração, segurança, portabilidade, entre outros. A nova versão do Debate já está sendo implementada através de um cliente HTML.

Tietze [2001] apresenta requisitos de usuário e de desenvolvedor que arquiteturas de groupware devem apresentar. A arquitetura proposta nesta tese atende aos requisitos de desenvolvedor, apresentados na Seção 2.1.1. Foi utilizada a linguagem de programação Java e uma extensão do modelo de componentes JavaBeans, ambos bem conhecidos e utilizados. Os *component frameworks* e os componentes encapsulam diversas complexidades técnicas de baixo nível e multidisciplinar, favorecendo o reuso da experiência e do conhecimento anterior (RD1). A centralização da arquitetura favorece o aproveitamento de modelos de dados existentes (RD2) e o compartilhamento transparente de dados (RD3). Dada a centralização da arquitetura cliente-servidor na web e a opção de não utilizar applets no ambiente, não há necessidade de suporte a dados locais (RD4). Alguns

dos componentes 3C disponibilizados tratam de informações de percepção, como o AwarenessMgr (RD5). Para disponibilizar novos serviços, basta implantar o arquivo correspondente na estrutura de diretórios definida (RD6). A escalabilidade é tratada pelos *component frameworks* e pelos frameworks de infra-estrutura (RD7). Para integrar ferramentas externas ao ambiente, é desenvolvido um adaptador e a ferramenta é empacotada no formato definido no modelo de componentes (RD8). Por fim, os *component frameworks* apresentam recursos para execução intermitente de serviços e funcionalidades no servidor (RD9). Os requisitos de usuários são dependentes da aplicação colaborativa construída.

Neste capítulo, foram apresentados a arquitetura e os problemas encontrados no desenvolvimento do AulaNet 2.0 e os novos requisitos e a arquitetura para o AulaNet 3.0. Os estudos de casos apresentados ilustraram a capacidade de composição, de re-configuração, de customização e de extensão da solução. A abordagem vem se mostrando apropriada para o re-desenvolvimento do ambiente AulaNet e de seus serviços. Foram também coletados indícios da aceitação da abordagem e da arquitetura propostas nesta tese por especialistas da área acadêmica e da empresa EduWeb. A abordagem também foi utilizada por alunos da disciplina Engenharia de Groupware como parte do trabalho necessário para aprovação. A abordagem e a arquitetura também foram utilizadas para instanciar o fórum de discussão do ambiente TelEduc e foram comparadas com a literatura. Apesar de nenhum dos indícios isoladamente possibilitar tirar conclusões, no conjunto, foram considerados satisfatórios.

6
Conclusão

Aplicações que estão disponíveis de forma monolítica são pouco flexíveis, visto que há muitas inter-relações e interdependências amarradas em seu código fonte. Isto dificulta a adaptação do sistema às preferências e necessidades dos usuários e à sua evolução. Um pacote de aplicações (*application suite*) oferece um conjunto de programas, normalmente de um mesmo fabricante, com algum nível de integração e uma aparência padronizada. Alguns exemplos de pacotes de aplicações são o Microsoft Office[30] e o Netscape Communicator suite[31]. Porém, nos pacotes não há muita flexibilidade (nem sempre o usuário pode decidir quais partes do pacote deseja instalar) e não é possível integrar ferramentas externas. Além disto, nestas aplicações não há necessariamente reuso e modularização do código. A versão 2.0 do AulaNet apresenta-se como um pacote de aplicações, oferecendo um conjunto de serviços relativamente independentes. Entretanto, seu código fonte é fortemente integrado.

A utilização de técnicas de desenvolvimento baseado em componentes é uma forma de desenvolver groupware extensível. Estas técnicas visam desenvolver sistemas modulares compostos de componentes de software, adaptáveis e combináveis na medida da necessidade, tendo em mente futuras manutenções. A utilização de componentes favorece a utilização de código já testado em outras situações e a captura e o encapsulamento do conhecimento e da experiência dos especialistas do domínio.

Greenberg [2006] argumenta que a utilização de toolkits com componentes que encapsulam as complexidades do desenvolvimento de groupware é uma maneira de impulsionar o desenvolvimento desta tecnologia, que ainda não atingiu seu potencial de disseminação e utilização, principalmente nas empresas. De acordo com Greenberg, uma parcela muito pequena dos groupwares

---

[30] http://office.microsoft.com
[31] http://browser.netscape.com

desenvolvidos tornou-se produtos comerciais realmente utilizados, apesar de sistemas para grupo terem sido previstos por Douglas Engelbart em 1968 e a área de CSCW ter sido estabelecida ao final da década de 80.

Ao analisar os motivos pelos quais a tecnologia de groupware ainda não deslanchou, Greenberg [2006] argumenta que diferentemente das aplicações desktop de automação de escritório e do hipertexto, que também foram previstas por Engelbart, o desenvolvimento de groupware não dispõe de um ferramental que simplifique a construção, aumente a produtividade e possibilite que programadores não tão experientes prototipem aplicações para grupos. As aplicações desktop e o hipertexto contam com toolkits e ambientes de desenvolvimento que estabelecem uma linguagem própria, utilizada pelos programadores para criar aplicações agrupando e interligando componentes. Com este ferramental, o foco do desenvolvimento é deslocado da criação de algoritmos referentes a problemas de baixo nível para a investigação da interação com o usuário e para o suporte computacional ao domínio em questão [Myers, 1995]. Tendo este ferramental disponível, uma grande quantidade de aplicações é criada, algumas comercialmente, outras por diversão e outras, por estudantes, como exercícios em seus cursos. Isto favorece a diversidade e a criatividade no desenvolvimento.

Ainda de acordo com o mesmo autor, atualmente o desenvolvimento de groupware ainda necessita de programadores altamente capacitados, aptos a lidar com protocolos de rede, soquetes, captura, apresentação, compressão e descompressão de áudio e vídeo, transações distribuídas, sincronização, concorrência de acesso, gerenciamento de sessão, entre outros. Com isto, mesmo questões bem conhecidas e investigadas pela comunidade de CSCW são deixadas de lado, pois o foco do desenvolvimento permanece no baixo nível e não há reuso do suporte computacional à colaboração. Nos sistemas colaborativos desenvolvidos desta maneira, o refinamento iterativo e a prototipação de novas idéias normalmente são dificultados, e o código fica fortemente acoplado. As ferramentas desenvolvidas acabam sendo pouco aderentes às reais necessidades de interação, que para serem descobertas requerem muita experimentação, investigação e prototipação. Greenberg [2006] posiciona o desenvolvimento de groupware no modelo BRETAM [Gaines, 1999], apresentado na Figura 6.1.
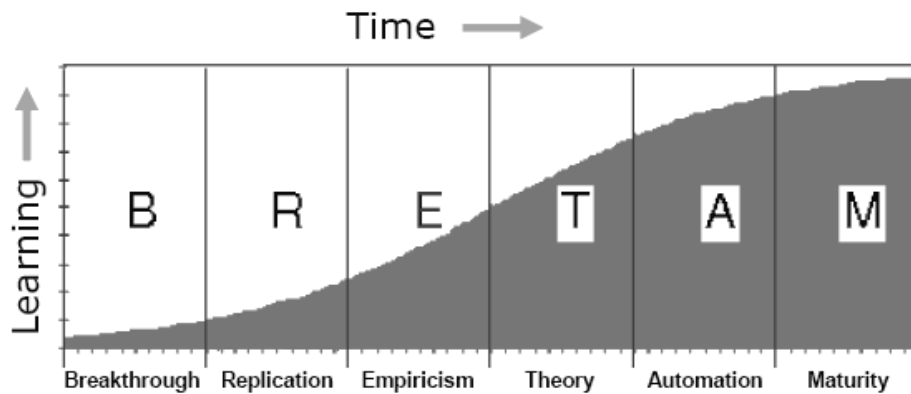
Figura 6.1. Modelo BRETAM para o desenvolvimento de uma tecnologia [Gaines, 1999]

O modelo BRETAM descreve como uma tecnologia evolui ao longo do tempo. A tecnologia inicia-se a partir de uma idéia criativa e uma visão de futuro (fase de *Breakthrough*). As idéias de groupware originaram-se da visão de Engelbart em 1968 e dos artigos seminais da década de 80. A fase de *Replication* ocorre quando as pessoas imitam as idéias umas das outras, reimplementando-as, alterando-as e inovando. Na replicação, estabelece-se uma comunidade de pesquisa, que adquire um corpo de conhecimento sobre os fatores centrais da tecnologia e sua relação com os seres humanos. O processo de construção das ferramentas ainda não é bem estabelecido e há muitas incertezas e retrabalho. De acordo com Greenberg [2006], groupware ainda está nesta etapa. Na fase de *Empiricism*, a tecnologia se torna bem disseminada e é adquirida uma larga experiência na aplicação da tecnologia a situações práticas e reais. Com base na experiência adquirida, são formuladas regras empíricas, que são descritas textualmente (boas práticas, padrões e guidelines) ou implementadas em ferramentas que as encapsulem e instrumentem o desenvolvedor. Ao ser adquirida mais experiência, teorias são desenvolvidas e validadas (fase de *Theory*) e, posteriormente automatizadas (fase de *Automation*). A fase de *Maturity* é atingida quando as tecnologias e teorias são utilizadas rotineiramente de forma transparente e sem questionamentos [Gaines, 1999].

A abordagem proposta nesta tese visa instrumentar o desenvolvedor de groupware com componentes concebidos com base no modelo 3C de colaboração. O desenvolvedor parte do levantamento de requisitos e da análise do domínio e seleciona os componentes necessários para oferecer suporte computacional à dinâmica estabelecida para a colaboração. Os conceitos abordados no modelo 3C

são usados para guiar a especificação do groupware e provêem uma linguagem comum para representar e descrever aspectos da colaboração.

Os conceitos da modelagem do domínio, que neste caso são referentes à colaboração, permeiam as diversas atividades e artefatos do desenvolvimento. A modelagem realizada é mapeada à implementação, suavizando a transição entre as atividades do desenvolvimento, o que favorece um ciclo iterativo e a manutenção do groupware. O modelo também propicia que no processo da engenharia sejam isolados os problemas referentes a cada C de modo a abordar cada elemento separadamente ao refinar o suporte à colaboração.

Greenberg [2006] propõe características, listadas na Tabela 6.1, que bons toolkits devem apresentar para instrumentar adequadamente o desenvolvedor de groupware e auxiliar a tecnologia a sair da fase de replicação do modelo BRETAM. A abordagem e a arquitetura propostas nesta tese atendem a estas características.

Num	**Característica**
1	Estar embutido em uma plataforma e utilizar linguagem familiar e de uso comum, de modo a facilitar a adoção solução e aproveitar conhecimentos e habilidades
2	Encapsular a complexidade de baixo nível inerente ao desenvolvimento de groupware, como transmissão de dados, compartilhamento, concorrência e gerenciamento de sessões
3	Minimizar a necessidade de tarefas rotineiras não essenciais
4	Encapsular projetos e conceitos bem sucedidos e conhecidos pela comunidade de pesquisa em módulos que possam ser incorporados ao software com pouco esforço de programação
5	Apresentar uma API concisa que encoraja as pessoas a pensar sobre groupware
6	Tornar as coisas simples alcançáveis com poucas linhas de código e tornar as complexas possíveis de serem feitas

Tabela 6.1. Características de bons toolkits [Greenberg, 2006]

A arquitetura e os componentes foram desenvolvidos na linguagem Java e utilizam uma extensão do modelo de componentes JavaBeans, que é largamente difundido e utilizado na literatura e comercialmente (característica 1). Várias das plataformas apresentadas no Capítulo 2 utilizam este modelo [Banavar et al., 1998; Marsic, 1999; Stiemerling et al., 1999; Chabert et al., 1998; Hummes & Merialdo, 2000]. A utilização de plataformas e tecnologias de suporte difundidas facilita a adoção da solução e sua curva de aprendizagem é reduzida. A infra-estrutura de execução, que é montada a partir de uma composição de frameworks de infra-estrutura [Barreto et al., 2005], e os *component frameworks* oferecem diversos serviços de sistema, o que alivia a necessidade de o desenvolvedor escrever código para implementar serviços de baixo nível (característica 2).

Alguns exemplos de serviços providos são persistência, transação declarativa, concorrência de acesso, comunicação remota, logging, instalação, atualização e remoção, autenticação, segurança, tratamento e direcionamento de requisições, validação, tratamento de erros, geração de relatórios, etc. A infra-estrutura também automatiza diversos procedimentos repetitivos do desenvolvimento e da aplicação (característica 3).

A solução encapsula projetos e conceitos bem sucedidos e conhecidos pela comunidade de pesquisa em módulos passíveis de serem incorporados ao software com pouco esforço de programação (característica 4). A abordagem e a arquitetura propostas nesta tese implementam e disponibilizam na forma de componentes diversos aspectos do modelo 3C de colaboração e possibilitam que novos conceitos, ou variações dos existentes, sejam disponibilizados em componentes, que são integráveis às ferramentas colaborativas. O conjunto de componentes disponibilizados apresenta funcionalidades de uma maneira coesa e fracamente acoplada, pela própria natureza dos componentes de software (característica 5). Por fim, Greenberg [2006] afirma que as adaptações simples devem ser conseguidas com poucas linhas de código (característica 6). Os estudos de caso apresentados no Capítulo 5 desta tese mostram que diversas adaptações são realizadas através de customizações ou substituições de componentes por versões mais robustas. Quando estas opções não forem suficientes, um novo componente é desenvolvido e, respeitando-se as interfaces definidas na arquitetura, incorporado e integrado aos componentes existentes.

Ao disponibilizar um conjunto de componentes fundamentados em um modelo de colaboração e construídos de modo a encapsular as dificuldades técnicas de sistemas distribuídos e multi-usuário, é estabelecida uma linguagem através da qual os desenvolvedores pensam e investigam o suporte computacional à colaboração [Whorf, 1956]. Com isto, a colaboração passa a estar presente nas diversas atividades e artefatos do desenvolvimento de groupware. De acordo com Grosz [1996], “Collaboration must be designed into systems from the start; it cannot be patched on” (a colaboração deve ser incorporada no projeto dos sistemas desde o início; não pode ser remendada).

A utilização de um modelo de colaboração e de componentes fundamentados neste modelo favorece a redução da distância semântica entre

modelagem e implementação, de modo que os desenvolvedores pensam em termos da colaboração nas diversas atividades do desenvolvimento. O sistema colaborativo é estruturado a partir de componentes que encapsulam e refletem os conceitos do modelo 3C, mapeando as necessidades da colaboração a arranjos de componentes. Este ferramental possibilita que programadores criem protótipos e investiguem suas idéias, copiando e variando idéias de outros, compondo novos ambientes e ferramentas colaborativas. O desenvolvedor, instrumentado pelos componentes pré-elaborados e pela infra-estrutura de execução, direciona seus esforços à investigação e projeto de um efetivo suporte à colaboração. O perfil deste desenvolvedor não necessita mais ser tão especializado em questões técnicas de baixo nível, o que é muito favorável ao desenvolvimento do ambiente AulaNet. De acordo com o diretor de tecnologia da empresa EduWeb, sr. Luidi Xavier Fortunato, é difícil de encontrar e caro de contratar especialistas na tecnologia de suporte.

O reuso obtido com componentes de software favorece a produtividade e a melhoria da qualidade [Pfleeger, 2001]. Boas soluções e inovações no suporte à colaboração são reusadas por diversos grupos de pesquisa e instituições, o que favorece o refinamento e amadurecimento dos componentes. O AulaNet é disponibilizado com um conjunto padrão de serviços que pode ser estendido para atender às necessidades específicas de cada grupo e situação, bem como à sua evolução.

6.1. Contribuições

A principal contribuição desta tese é a proposta de utilização do modelo 3C de colaboração, originado de um artigo seminal de Ellis et al. [1991], freqüentemente citado e utilizado na literatura, para embasar o desenvolvimento de groupware, instrumentando o desenvolvedor com kits de componentes fundamentados no modelo. Conforme discutido no Capítulo 2 desta tese, esta abordagem é inovadora, e, conforme discutido no Capítulo 5, vem se mostrando viável e tem encontrado boa aceitação nos veículos científicos da área e de áreas afins.

Outra contribuição deste trabalho é o aprofundamento e refinamento do modelo 3C. Diversos trabalhos da literatura e ferramentas utilizadas no meio acadêmico e comercial foram estudados com este propósito. Além disto, o modelo 3C foi consolidado a partir da experiência adquirida nos 8 anos de desenvolvimento e utilização do ambiente AulaNet no suporte computacional à colaboração.

Outra contribuição deste trabalho é a proposição de uma arquitetura baseada em component frameworks e component kits para a construção de groupware. O próprio Collaboration Component Kit é uma contribuição deste trabalho. Como contribuição específica para o projeto AulaNet foram desenvolvidos sua nova arquitetura, com base na arquitetura proposta, e foram desenvolvidos dois de seus serviços. Apesar dos demais serviços não terem sido implementados, eles foram considerados no projeto e implementação da arquitetura. Com base no component kit proposto, os outros serviços serão construídos.

Por fim, o trabalho desenvolvido nesta tese deu origem a diversas publicações e a um curso do Departamento de Informática da PUC-Rio.

6.2. Limitações

A abrodagem e o ferramental propostos nesta tese não se aplicam a qualquer groupware. A dinâmica da colaboração e os serviços colaborativos devem ser bem modelados pelo modelo 3C. A aboragem e o ferramental não são voltados a princípio para a implantação ou refinamento do suporte a colaboração em sistemas legados.

Nesta tese, só é tratada a camada de negócio do desenvolvimento de groupware. O desenvolvimento de groupware engloba muitas outras atividades e fatores. Por exemplo, a infra-estrutura de execução e a interface com o usuário também são cruciais para a criação de groupware propício para utilização e manutenção. A camada de infra-estrutura é tratada na dissertação de Celso Gomes Barreto [2006], que é parte deste consórcio de pesquisa. A camada de interface foi tratada apenas na arquitetura técnica, devendo ser investigada na arquitetura de aplicação.

Os estudos de caso desta tese foram restritos às ferramentas de comunicação Conferências e Debate. Mesmo estas ferramentas possuindo diversos elementos de coordenação e cooperação, além dos de comunicação, é necessário avaliar e refinar a engenharia do domínio para ferramentas de coordenação e de cooperação para verificar a abrangência do trabalho.

As dificuldades de aplicar groupware não são tratadas diretamente neste trabalho. Há diversos fatores envolvidos, como questões políticas, conflitos, intrigas, falta de estímulo, individualismos, falta de afinidade, questões culturais, etc. A composição do grupo de trabalho, por exemplo, tem uma influência direta em seu sucesso [Roussos et al., 1997]. Romano et al. [1998] relata que a presença de indivíduos controladores tende a desestabilizar a colaboração, e Borges et al. [2002] investiga os papéis informais que os participantes desempenham em uma reunião, de modo a compor grupos mais efetivos. A dinâmica de trabalho, as ferramentas selecionadas e a composição do ambiente de trabalho são fundamentais para propiciar a colaboração. Estes fatores são especialmente críticos em um ambiente voltado para o ensino-aprendizagem. Neste tipo de ambiente, normalmente não é possível escolher o grupo de trabalho: o grupo é composto pelos alunos que se matriculam no curso. Além disto, os alunos não estão habituados e capacitados a colaborarem e os professores não estão habituados e capacitados a planejarem dinâmicas que explorem a colaboração nas atividades educacionais. A própria utilização de ambientes educacionais pela web ainda é uma barreira a ser vencida [Lucena & Fuks, 2000].

Outra limitação desta tese é que o modelo 3C não foi investigado em grupos numerosos. Os grupos tratados possuíam menos de 20 participantes. De acordo com Teles [2004] a colaboração é dificultada em grupos numerosos, pois os subgrupos, as vaidades, as inimizades, etc. se intensificam, e é mais difícil para um coordenador organizar, sincronizar e unir o grupo. O suporte computacional à colaboração proposto nesta tese pressupõe equipes de trabalho reduzidas e coesas.

Este trabalho não foi comparado em termos de qualidade de software com outras abordagens. A qualidade de software é tratada pela norma ISO/IEC 9126 e sua equivalente brasileira NBR 13596, que estabelece seis características de qualidade de software: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Alguns destas características são dependentes

da infra-estrutura de execução e da arquitetura técnica adotada. A abordagem proposta nesta tese influencia mais diretamente as características funcionalidade, confiabilidade e manutenibilidade, principalmente as subcaracterísticas adequação, maturidade e modificabilidade, respectivamente. A adequação é favorecida pela disponibilização de serviços e componentes 3C, que propiciam a construção de um groupware mais ajustado às necessidades específicas de colaboração do grupo. A maturidade é favorecida pelo reuso dos componentes de software e pelo encapsulamento do suporte à colaboração. A modificabilidade é favorecida pela modularização e pela capacidade de substituição obtida com a componentização. Para comparar em termos de qualidade a abordagem proposta nesta tese com outras abordagens disponíveis na literatura, seria necessário que grupos externos desenvolvessem o mesmo ambiente, o AulaNet, por exemplo, utilizando as várias abordagens, para depois comparar os resultados. Por restrições de tempo e de recursos, isto não foi tratado no escopo desta tese. Algumas comparações foram feitas com o AulaNet 2.0 e o AulaNet 3.0. Contudo, os dois ambientes foram desenvolvidos em tempos diferentes, por desenvolvedores com perfis diferentes e com tecnologias e ferramentas diferentes, o que impossibilita extrair afirmações conclusivas. Além disto, não foi utilizada uma abordagem orientada para o reuso no AulaNet 2.0.

6.3. Trabalhos Futuros

Pretende-se investigar os aspectos relacionados à construção da interface com o usuário, levando em consideração o modelo 3C de colaboração e a arquitetura proposta nesta tese. Eventualmente, um mesmo componente 3C pode ser associado a mais de um widget de interface. Por exemplo, o componente CategorizationMgr, pode ser associado a um widget que monta a lista de categorias em um SelectBox ou a um widget que disponibiliza a lista de categorias através de Radio Buttons. Esta variedade de componentes de interface organizados em função dos componentes 3C favorece o desenvolvedor, que passa a contar com palhetas de componentes para seleção do que for mais apropriado para suas necessidades. Chung & Dewan [2004] propõem uma abordagem para

mapeamento dinâmico de componentes de interface com o usuário e de negócio, que pode ser levada em consideração nesta extensão da solução.

Os trabalhos futuros desta tese também incluem uma investigação e experimentação maior sobre qualidade de software no escopo da abordagem proposta e a investigação da utilização da solução apresentada em grupos numerosos e heterogêneos. Pretende-se também desenvolver adaptadores do modelo de componentes utilizado no AulaNet para as infra-estruturas de execução dos portalware, de modo a compatibilizar os serviços do AulaNet com os portalware e vice-versa.

Pretende-se também manter a experimentação com a turma da disciplina de Engenharia de Groupware, que utiliza a abordagem e a arquitetura propostas nesta tese para projetar uma extensão ou um novo groupware. Nas próximas edições do curso a documentação dos componentes será aprimorada e os alunos implementarão as propostas, utilizando o ferramental provido. Estas novas experimentações possibilitarão refinar o conjunto de componentes 3C e sua documentação. Para guiar os alunos no desenvolvimento da solução, será utilizado o processo proposto por Pimentel [2006], visando direcioná-los e reduzir a quantidade de falhas.

Os componentes 3C também serão utilizados pelo grupo de desenvolvedores do AulaNet do Laboratório de Engenharia de Software (LES) da PUC-Rio para implementar os demais serviços do ambiente. Depois de implementados, estes serviços serão entregues para o grupo de desenvolvedores da EduWeb, que irá trabalhar no código. Destes desenvolvimentos, espera-se também o refinamento do ferramental proposto.

Como trabalho futuro, conforme discutido no Capítulo 4, também pretende-se desenvolver e investigar a utilização de frameworks de domínio para a geração de aplicações, serviços e componentes de uma mesma família, com o objetivo de aumentar o reuso de código, facilitar a instanciação de componentes similares e consolidar e generalizar o suporte à colaboração.

Um problema ainda em aberto no desenvolvimento baseado em componentes é a construção de efetivos mecanismos de localização de componentes [Gimenes & Huzita, 2005]. O grau de reuso não cresce com o

aumento do número de componentes disponíveis, em parte pela dificuldade de localizá-los [Sametinger, 1997]. Pretende-se investigar como criar uma comunidade de desenvolvimento e de troca de componentes para o AulaNet, definindo maneiras de catalogar e buscar componentes, utilizando os recursos de catálogos e localização de serviços da plataforma web services.

Outro fator a ser investigado é a utilização da tecnologia de aspectos para capturar e explicitar os pontos de variabilidade dos componentes [Kulesza et al., 2004]. Pretende-se investigar também o tratamento do versionamento e o controle de dependências entre versões dos componentes. Os trabalhos futuros também estão direcionados para a criação de um ambiente voltado para o trabalho em grupo e para a definição de uma engenharia de groupware.

6.3.1. O Ambiente eLabora

O conceito de trabalho está mudando, em parte devido ao ritmo de produção de conhecimento e ao aperfeiçoamento das tecnologias de telecomunicações. Profissionais dedicados ao trabalho intelectualizado são cada vez mais requisitados. Destes trabalhadores são exigidas novas habilidades: eles devem estar aptos a aprender continuamente, a trabalhar em grupo e a transformar criativamente conhecimento em novo conhecimento. Estas necessidades definem uma situação onde seus ambientes de trabalho e de aprendizado se confundem e se misturam [Fuks, 2000].

Pretende-se utilizar a experiência e a infra-estrutura desenvolvida para o AulaNet para a construção de um ambiente voltado para o trabalho em grupo, principalmente para o suporte a projetos colaborativos, onde normalmente há uma rotatividade de participantes e pouco suporte computacional. Este ambiente, já batizado de eLabora [Gerosa et al., 2001], integra as necessidades do ensino-aprendizagem e do trabalho. Com algumas adaptações, principalmente na nomenclatura, é possível imaginar que um curso do AulaNet é correspondente a um projeto do eLabora, uma turma a uma equipe, uma aula a uma etapa de trabalho, etc.

Com o eLabora, espera-se capacitar trabalhadores para enfrentar os desafios da sociedade do conhecimento, trabalhando em um ambiente similar ao que são treinados. O desenvolvimento do eLabora auxiliará também a refinar a arquitetura e os serviços propostos. Além disto, por utilizar a mesma infra-estrutura, um serviço colaborativo desenvolvido para o AulaNet poderá ser disponibilizado no eLabora e vice-versa.

6.3.2. A Engenharia de Groupware Baseada no Modelo 3C

Conforme discutido no Capítulo 2, há diversas abordagens na literatura que estendem as ferramentas e técnicas da Engenharia de Software de modo específico para o desenvolvimento de groupware. Esta tese está inserida em um contexto de pesquisa que objetiva descrever uma engenharia de groupware baseada no modelo 3C e em técnicas de desenvolvimento baseado em componentes, instrumentando as diversas atividades do desenvolvimento de aplicações colaborativas. A engenharia de groupware é o processo sistemático, disciplinado e instrumentado pelo qual se modela e se desenvolve groupware [Fuks et al., 2002]. A engenharia de groupware une conceitos e tecnologias das áreas de Engenharia de Software e de CSCW para instrumentar o desenvolvimento e evolução de groupware.

Idealmente, um groupware deve ser prototipado [Schrage, 1996], dado sua tendência a falhas [Grudin, 1989]. Por isto, um ciclo baseado no desenvolvimento incremental [Boehm, 1988] mostra-se mais adequado para o desenvolvimento de groupware. Este ciclo é especialmente útil em projetos com requisitos instáveis ou não bem-definidos, que é justamente o caso de groupware. O desenvolvimento incremental com sucessivos testes de aceitação e de usabilidade adapta o groupware na direção das reais necessidades do usuário. Como são necessárias várias iterações e as necessidades não se estabilizam, visto que o grupo continuamente muda suas características, o desenvolvimento baseado em componentes mostra-se uma solução para propiciar a prototipação rápida, a experimentação e a adaptação do suporte computacional à colaboração. A Figura 6.2 ilustra o ciclo de desenvolvimento da engenharia de groupware proposta [Fuks et al., 2005].
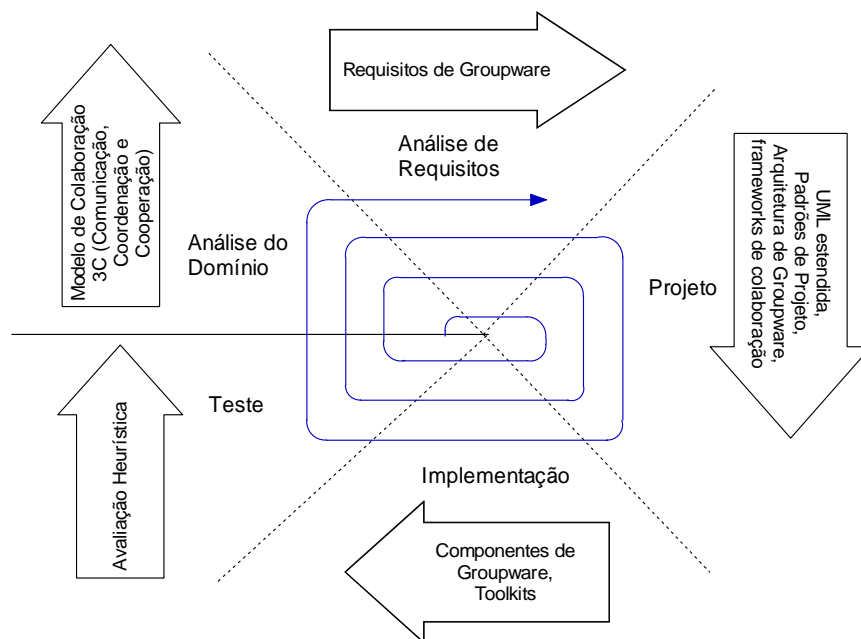
Figura 6.2. Ciclo da engenharia de groupware

O ferramental de suporte das diversas atividades do ciclo de desenvolvimento de groupware é embasado no modelo 3C, o que propicia trabalhar com os mesmos conceitos desde os primeiros momentos de análise até os testes finais, valorizando o desenvolvimento iterativo. Mantendo os mesmos conceitos na análise do domínio, na análise de requisitos, no projeto, na implementação e nos testes cria-se um vocabulário comum, mapeando conceitos e termos da realidade para a implementação, diminuindo a distância semântica e favorecendo a evolução da aplicação.

Considerando a análise do domínio como “o processo de identificar, coletar, organizar e representar a informação relevante de um domínio com base no estudo de sistemas existentes, conhecimento capturado de especialistas do domínio, fundamentação teórica e tecnológica” [Hess et al., 2000], considera-se que o modelo 3C instrumenta o desenvolvedor, que analisa a aplicação de sua ferramenta com base nos conceitos referentes à comunicação, à coordenação e à cooperação. A análise do domínio é utilizada como base para a definição e a especificação dos requisitos e é onde são identificados e organizados os conhecimentos sobre uma classe de problemas [Werner & Braga, 2005].

O levantamento de requisitos é guiado por requisitos genéricos de groupware e pelos requisitos de colaboração levantados durante a análise baseada no modelo 3C. Os requisitos, entretanto, nunca são precisos o suficiente para

possibilitar uma especificação precisa do sistema. O desenvolvimento incremental possibilita avaliar e validar continuamente os requisitos, bem como as decisões de projeto e de implementação. O desenvolvimento passado ajuda a entender o desenvolvimento futuro [Teles, 2004].

Há diferentes técnicas baseadas no modelo 3C que instrumentam o projeto de um groupware. Por exemplo, padrões, para reusar abordagens comuns de projeto e de análise de groupware baseado em componentes 3C; extensões da UML, para representar aspectos específicos de groupware e da colaboração direcionando a integração com a arquitetura; arquiteturas de groupware baseada em c*omponent frameworks*; e frameworks de aplicação para instanciar ambientes, serviços e componentes 3C.

Para a fase de implementação, componentes pré-elaborados de comunicação, de coordenação e de cooperação podem ser plugados nos *component frameworks*, como ilustrado nos estudos de caso desta tese. São disponibilizadas também ao desenvolvedor ferramentas que auxiliam a montagem e customização, como a apresentada por Stiemerling et al. [2001].

Os requisitos e conceitos do modelo 3C também guiam a validação e os testes heurísticos do groupware, focando a atenção do avaliador em aspectos específicos de comunicação, de coordenação e de cooperação. Além das técnicas de avaliação heurística, são utilizadas técnicas tradicionais da Engenharia de Software, como testes unitários, para buscar a presença de erros no sistema.

O ciclo de desenvolvimento de groupware está inserido no contexto de um processo de desenvolvimento de groupware. O processo define as atividades e os artefatos a serem manipulados durante o desenvolvimento, que no caso da engenharia de groupware proposta, são baseados no modelo 3C de colaboração [Pimentel et al., 2005]. O processo prevê a instanciação da solução e a criação de novos componentes.

6.4.
Considerações Finais

A abordagem proposta nesta tese objetiva instrumentar o desenvolvedor de groupware para que ele, a partir das necessidades do trabalho em grupo, estruture

a aplicação colaborativa utilizando componentes fundamentados no modelo 3C de colaboração. Deve ser lembrado, porém, que como apontado por Laurillau & Nigay [2002], trazer a separação do suporte computacional dos diversos elementos da colaboração para a interface com o usuário pode trazer confusão e falhas de entendimento. Os elementos da interface com o usuário devem ser harmoniosamente combinados e posicionados próximos aos objetos que afetam.

Vale ressaltar também que a arquitetura componentizada do AulaNet não garante por si só a qualidade do produto final. O mesmo ocorre com um restaurante, cujo sucesso depende mais do desempenho do cozinheiro do que da qualidade da panela usada para o preparo da comida, dos ingredientes ou do prato no qual ela é servida. O sucesso do groupware é dependente de quem monta o ambiente e do desempenho de quem atua nele. Não basta conhecimento de programação para se projetar um groupware, são necessários também conhecimentos sobre o funcionamento da colaboração em um grupo de trabalho e dos requisitos de usuários e de desenvolvedores. O processo desenvolvido na tese de Mariano Pimentel ameniza esta dependência, pois para desenvolver groupware, o desenvolvedor segue os passos e gera os artefatos pré-estabelecidos [Pimentel et al., 2005].

Espera-se com a abordagem e a arquitetura propostas uma maior manutenibilidade e adaptabilidade do AulaNet. Espera-se, com isto, que a evolução do ambiente não desestruture o código, que passa a ser construído com base nos pilares do modelo 3C: comunicação, coordenação e cooperação.

Apêndice A
Componentes e Frameworks

Para atingir a qualidade e cumprir o orçamento e os prazos previstos, não é mais possível começar a construir software a partir do zero e estruturá-lo na forma de blocos monolíticos onde uma modificação propaga efeitos colaterais por todo o código, dificultando a manutenção e a substituição de suas partes [Werner & Braga, 2005, p.66]. O Desenvolvimento Baseado em Componentes (DBC) surgiu como uma abordagem para o desenvolvimento de software, cujo objetivo é a quebra dos blocos monolíticos em componentes interoperáveis, instrumentando o desenvolvimento e reduzindo seus custos, por meio do reuso de componentes [Sametinger, 1997]. O software passa a ser composto de partes relativamente independentes, que foram concebidas para serem substituíveis, reusáveis e interoperáveis em uma infra-estrutura de execução específica.

A.1.
Componentes de Software

Componentes têm um papel de destaque em outras engenharias, uma vez que permite a adoção do conceito de "caixa-preta", que possibilita ao desenvolvedor um maior nível de abstração e independência. Na indústria automobilística, é possível utilizar o mesmo pneu, parafuso, gasolina e motor para diferentes marcas e modelos de automóveis. Não é necessário, e seria demasiadamente caro, reprojetar e construir sob demanda cada peça para cada carro.
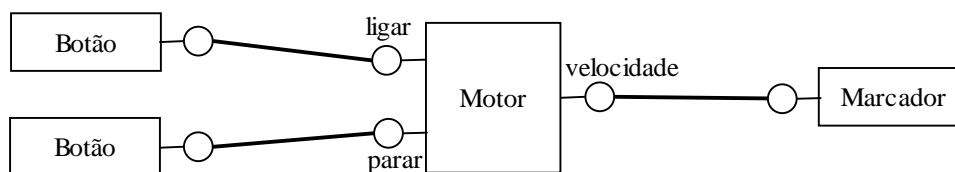
Figura A.1. Exemplo de uso de componentes [D'Souza & Wills, 1998, p.405]

O exemplo da Figura A.1, tratado por D'Souza & Wills [1998], ilustra a componentização na ligação de um motor a dois botões e a um marcador. Os mesmos botões e o marcador são reusáveis em outras situações (por exemplo, utilizando o marcador para marcar a temperatura) e estes componentes são substituíveis por outros equivalentes, porém com implementações diferentes (por exemplo, substituindo os dois botões por uma chave liga-desliga ou o marcador analógico por um digital). A interoperabilidade é propiciada pela compatibilidade entre os conectores. O conjunto todo pode ser encarado como um componente de um sistema maior.

A componentização de software visa trazer estes princípios para o desenvolvimento: reuso, substituição e montagem [D'Souza & Wills, 1998]. O desenvolvedor passa a desenvolver pedaços de software encapsulados na forma de componentes para que outros desenvolvedores possam utilizá-los, substituí-los ou modificá-los, com efeitos colaterais reduzidos.
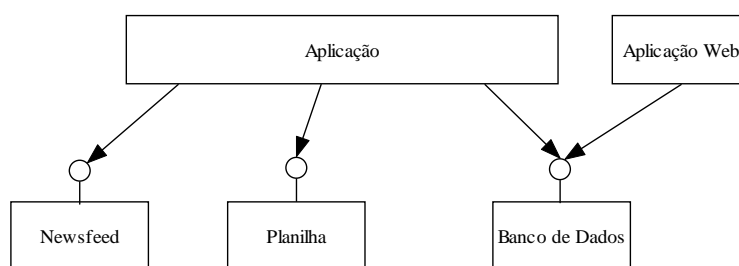
Figura A.2. Exemplo de uso de componentes de software [D'Souza & Wills, 1998, p.384]

D'Souza & Wills [1998] exemplificam a composição de software através da situação ilustrada na Figura A.2. Uma aplicação lê dados de uma fonte sobre ações, registra em uma planilha e passa os resultados a um banco de dados. O banco de dados é compartilhado por uma aplicação web, que extrai informações sob demanda. Cada componente é relativamente independente, compartilhado e utilizado para compor um sistema maior. Os componentes disponibilizam interfaces para interconexão e são substituíveis por outros equivalentes. Eventualmente, um componente não precisa de interação com o usuário ou de mecanismos de persistência.

A.2.
Definição de Componente de Software

Nesta seção são apresentadas algumas definições para componente de software encontradas na literatura. O termo componente também é comparado com outros termos e conceitos utilizados no desenvolvimento de software. Para alguns autores um componente de software é qualquer elemento reusável: código binário, código fonte, estruturas de projeto, especificações e documentações [Krueger, 1992]. Outros autores focam na tecnologia, e consideram como componente qualquer pedaço de código que segue uma especificação. A Tabela A.1 apresenta algumas definições encontradas na literatura, ordenadas cronologicamente.

Booch [1987]:	Módulo logicamente coerente e fracamente acoplado que denota uma abstração única.
Component Int. Labs [Orfali et al., 1995]:	Pedaço de software pequeno o suficiente para implementar e manter, grande o suficiente para distribuir e dar suporte, e com interfaces padronizadas para oferecer interoperabilidade.
Brown [1996]:	Parte não-trivial de um sistema, praticamente independente e substituível, com uma função clara no contexto de uma arquitetura bem definida.
Sametinger [1997]:	Pedaço de software autocontido, claramente identificável, que descreve ou executa funções específicas, tem interfaces claras, documentação apropriada e um status de reuso.
Szyperski [1997, p.34]:	Unidade binária com interfaces contratualmente especificadas e dependências de contexto explícitas, instalável de forma independente e usado por terceiros sem modificação para compor aplicações finais.
D'Souza & Wills [1998, p.387]	Um pacote coerente de software que (a) pode ser desenvolvido e instalado independentemente como uma unidade, (b) tem interfaces explícitas e bem definidas para os serviços que provê, (c) tem interfaces explícitas e bem definidas para os serviços que espera de outros, e (d) pode ser utilizado para composição com outros componentes, sem alterações em sua implementação, podendo eventualmente ser customizado em algumas de suas propriedades.
Councill & Heineman [2001, p.7]	Um *componente de software* é um elemento que está em conformidade com um modelo de componentes e pode ser instalado independentemente e composto sem modificações.
Barroca et al. [2005]:	Unidade de software independente, que encapsula, dentro de si, seu projeto e implementação, e oferece serviços, por meio de interfaces bem definidas, para o meio externo.
OMG [2005]:	Um componente representa uma parte modular de um sistema que encapsula seu conteúdo e cuja manifestação é substituível. Um componente define seu comportamento através de interfaces fornecidas e requeridas.

Tabela A.1. Definições de componente de software

Neste trabalho, é adotada a definição proposta por D'Souza & Wills [1998], que enfatiza a possibilidade de customização e a necessidade de explicitar as interfaces fornecidas e requeridas. Também são consideradas as definições de

Szyperski [1997], que diz que um componente é uma unidade executável[32], e a de Councill & Heineman [2001, p.7], que diz que o componente segue um modelo de componentes. Apesar destas duas características não estarem explícitas na definição proposta por D'Souza & Wills [1998], os autores as adotam em seu livro. Das outras definições apresentadas na tabela, algumas são razoavelmente equivalentes à definição adotada [Sametinger, 1997; Orfali et al., 1995; Barroca et al., 2005], e outras são mais genéricas [Booch, 1987; Brown & Wallnau, 1996; OMG, 2005].

Componentes de software são utilizados nas mais diversas áreas e com os mais diversos propósitos. Um exemplo bastante conhecido é o uso de plugins nos navegadores web, utilizados para estender o suporte à visualização de conteúdos, como animações flash, documentos Word e vídeos diversos. Os plugins se comportam como componentes, pois são autocontidos, reusáveis, substituíveis, provêem serviços específicos de uma maneira coesa e bem definida, seguem uma padronização e são disponibilizados como uma unidade executável em uma plataforma pré-definida. Várias aplicações utilizam o suporte a plugins para estender as funcionalidades nativas, como o Adobe Photoshop, o Eclipse IDE e o Apache Web Server.

Componentes visuais de interface com o usuário (widgets) é outro exemplo bastante difundido do uso de componentes. Alguns ambientes de desenvolvimento integrado (IDEs) disponibilizam ao desenvolvedor um conjunto de componentes visuais que são utilizados para compor a interface com o usuário. O programador instancia e posiciona os componentes, configura suas propriedades e associa métodos a seus eventos, sem ter acesso a seu código fonte.

Dada a elasticidade e o desgaste do termo componente, é apropriado diferenciá-lo de outros conceitos e tecnologias para melhor caracterizá-lo. A seguir, o conceito de componente de software é comparado com os conceitos de módulo, objeto, classe, biblioteca e API.

---

[32] Clemens Szyperski, na segunda edição de seu livro "Component Software – Beyond Object-Oriented Programming", lançada em 2002, passou a caracterizar um componente como executável, ao invés de binário, visto que a capacidade de execução em uma plataforma é mais relevante do que a forma de empacotamento, afirmando que eventualmente um componente é implementado na forma de um script de código.

O conceito de componente é similar ao conceito tradicional de módulo, presente há bastante tempo na Engenharia de Software. A modularidade é de fato um pré-requisito para a tecnologia de componentes [Szyperski, 1997, p.33], que é diferenciada pela existência de uma padronização (*component model*) e de uma infra-estrutura específica para seu gerenciamento e execução (*component framework* e *container*) [D'Souza & Wills, 1998, p.386].

Também há uma certa confusão entre componentes e objetos, principalmente porque a maioria dos componentes é implementada utilizando linguagens orientadas a objetos [D'Souza & Wills, 1998, p.390]. O componente cria, envia e recebe objetos, e muitas vezes é representado e acessado por meio deles.

Componente e classe também são conceitos relacionados. Ambos utilizam polimorfismo e ligação dinâmica, realizam interfaces, podem participar de um relacionamento de dependência e composição, admitem instâncias e são participantes de interações [Booch et al., 2000]. Entretanto, classe e componente estão em níveis de abstração diferentes. Classe é uma abstração lógica do domínio (classe conceitual) ou uma estrutura de uma linguagem de programação utilizada para instanciar objetos (classe de software). Um componente é uma unidade executável, que pode ser a implementação "física" de uma ou mais classes[33]. Uma classe só pode pertencer a um único componente [Szyperski, 1997, p.32].

Um componente não necessariamente precisa estar na mesma máquina que a aplicação que o utiliza e nem escrito na mesma linguagem de programação. Diferentemente de classes, um componente pode estar disponível somente na forma binária e o nome do componente não pode ser utilizado para definir o tipo de uma variável ou parâmetro [Weinreich & Sametinger, 2001, p.36]. As padronizações para componentes também são mais abrangentes do que as padronizações para classes, definindo, entre outros fatores, empacotamento, ciclo de vida, conectores, interfaces providas e requeridas, etc. [D'Souza & Wills, 1998, p.390]. Componentes também têm uma gama maior de mecanismos de intercomunicação, como eventos e workflow, além das mensagens da orientação a

[33] Um componente não obrigatoriamente precisa conter classes [Szyperski, 1997, p.31]. Um componente pode conter código escrito em outras tecnologias.

objetos. Por fim, instâncias de componentes tendem a ser mais estáticas do que instâncias de classes, não tendo sua configuração alterada ao longo do ciclo de vida [D'Souza & Wills, 1998, p.391]. Mesmo estando em níveis de abstração diferentes, por simplificação do discurso, costuma-se chamar de componente a(s) classe(s) utilizadas na construção do componente.

Biblioteca de funções é outro termo similar a componente. Uma biblioteca provê serviços coesos para um sistema e é autocontida, reusável e substituível. Uma biblioteca pode ser disponibilizada na forma binária e orientada a objetos. Entretanto, uma biblioteca normalmente não pressupõe uma padronização, um modelo específico, um ciclo de vida, instalação e configuração (*deploy*), e nem a existência de uma plataforma específica de execução.

Um componente, assim como uma biblioteca, provê uma API (*Application Program Interface*), que representa parte do "contrato" de utilização daquele pedaço de código. Uma API expõe o conjunto de operações, procedimentos, funções, tipos e constantes que um pedaço de código oferece para ser utilizado externamente. Respeitando a API e os requisitos não-funcionais, é possível trocar o componente ou a infra-estrutura, sem impactar o código cliente. Por exemplo, o sistema operacional Windows oferece uma API padrão que é compatível entre suas diversas versões, de modo que um software desenvolvido para uma versão anterior do sistema operacional normalmente continua a funcionar em uma versão mais recente. O ambiente de trabalho é composto instalando e configurando os componentes necessários.

Muitas vezes, o termo componente é usado com pouco rigor em diversos níveis de abstração, o que pode ser a origem da confusão sobre o termo [D'Souza & Wills, 1998, p.388; Apperly, 2001, p.29]. Costuma-se chamar de componente: a especificação do componente, a implementação (código fonte), o executável que implementa a especificação (*deployable component*), a instalação em particular daquele executável, a execução específica daquele executável e a instância do componente. Com relação à granularidade, um componente de software pode ser construído a partir de uma única classe ou ser tão complexo como um subsistema [Gimenes & Huzita, 2005]. As definições apresentadas na Tabela A.1, ajudam a dar indícios sobre os critérios para considerar um pedaço de código como sendo um componente: ele deve ser passível de implementar e manter

independentemente, coeso, não-trivial e com uma função clara no contexto da arquitetura, bem como passível de separação, distribuição e reuso.

Resumindo, no contexto do DBC, componente de software é um elemento arquitetural mapeado em um arquivo executável, que segue uma especificação, e foi concebido para ser autocontido, reusável, substituível, além de prover serviços específicos de uma maneira coesa e bem definida. Portanto, classificar algo como componente depende também de como o código foi concebido e construído e como ele se relaciona com o restante do sistema, e não somente da aderência a uma padronização. Vale ressaltar que na literatura costuma-se encontrar alguns sinônimos para componentes, utilizados em contextos específicos, como plugin, add-on, módulo, serviço e widget.

A.3. Utilização de Componentes

Nesta seção são definidos alguns termos referentes à utilização de componentes (porta, conector, interface, adaptador, instância e *deployment*). Também são discutidas as maneiras de customizar componentes.

Uma **porta** é um meio identificável de conexão, por onde um componente oferece seus serviços ou acessa os serviços dos outros [OMG, 2005]. Cada porta tem uma identificação (nome ou número pelo qual é acessada). Cada porta define os tipos de valores que são transmitidos ou recebidos e agrupa interfaces [D'Souza & Wills, 1998, p.415].

Ao meio por onde é feita a conexão entre duas ou mais portas é dado o nome de **conector** [D'Souza & Wills, 1998, p.414; OMG, 2005]. O conector é implementado por invocação explícita de função, envio de mensagens síncronas ou assíncronas, propagação de eventos, stream de dados, workflow, código móvel, diálogo através de uma API, transferência de arquivo, pipe, propagação de eventos, buffer, sinalização, compartilhamento de arquivo via ftp, etc. [D'Souza & Wills, 1998, p.389, p.395; Wills, 2001, p.312]. Os tipos de conectores variam para cada tecnologia e ferramenta adotada. Por exemplo, JavaBeans oferece um conjunto de conectores diferentes dos oferecidos pelo COM+ [D'Souza & Wills, 1998, p.414]. Tipicamente, os componentes interagem entre si através de

chamadas de métodos, em um estilo cliente/servidor ou publicador/ouvinte [Weinreich & Sametinger, 2001, p.43]. A conexão entre os componentes é feita em tempo de codificação, compilação, inicialização ou execução. O conector além do fluxo de informações define também o fluxo de controle [Wills, 2001, p.313].

A **interface** de um componente define seus pontos de acesso, por meio dos quais outros componentes utilizam os serviços oferecidos [Szyperski, 1997, p.40]. A interface representa o contrato de utilização do componente. Respeitando os contratos, pode-se alterar a implementação interna do componente ou substituí-lo[34] por outro, sem modificar quem o utiliza. O contrato cobre aspectos funcionais e não funcionais [Szyperski, 1997, p.370]. Aspectos funcionais incluem a sintaxe e a semântica da interface. Os não funcionais incluem os aspectos referentes à qualidade de serviço.

A interface de um componente de software funciona como a especificação dos pinos de um circuito integrado. Para usar o circuito, basta conhecer o funcionamento de sua interface externa e prever na arquitetura do circuito, o encaixe para ele. Não é necessário o conhecimento dos detalhes internos de funcionamento do componente (abordagem caixa preta). Ao estabelecer o contrato de utilização entre o componente e seus clientes e separar a especificação e a implementação do componente, os componentes são desenvolvidos e substituídos transparentemente para seus clientes [Szyperski, 1997, p.34].

A interface define os serviços que os clientes podem requisitar de um componente e as restrições que devem ser observadas pelos componentes e clientes [Councill & Heineman, 2001, p.8; Weinreich & Sametinger, 2001, p.39]. Um componente apresenta múltiplas interfaces correspondendo aos conjuntos de serviços que visam diferentes necessidades dos clientes, de modo a simplificar o uso e reduzir o acoplamento entre os componentes [D'Souza & Wills, 1998, p.397]. As interdependências entre os componentes ficam restritas às interfaces específicas em vez de abranger o componente como um todo [OMG, 2005]. A substituição é realizada levando em conta somente as interfaces utilizadas.

---

[34] Para um componente substituir outro, o substituto deve prover ao menos os mesmos serviços solicitados ao original e não deve solicitar serviços distintos dos que o anterior utilizava [D'Souza & Wills, 1998].

Normalmente, um componente possui pelo menos duas interfaces. Uma interface é relativa à funcionalidade que o componente oferece a outros componentes (interface de negócio) e outra à conexão com a infra-estrutura de execução (interface de sistema). Na conexão com a infra-estrutura de execução são tratados serviços técnicos, como os relacionados ao ciclo de vida, à instalação e à persistência. Apperly [2001, p.30] compara componentes com janelas utilizadas na construção civil. As janelas são entregues como um pacote fechado, são plugadas em uma infra-estrutura, com um determinado propósito, e suas funcionalidades são utilizadas por seus clientes. Há uma interface que define a utilização pelos clientes (interface de negócio) e uma outra interface que determina a junção da janela com a infra-estrutura, especificando quantos e quais buracos de fixação são necessários (interface de sistema). Tanto o cliente quanto a infra-estrutura são independentes dos detalhes internos de implementação (revestimento dos pregos, tipo de cola empregada, etc.).

As interfaces são fornecidas (*provided interfaces*) ou requeridas (*required interfaces*) [Councill & Heineman, 2001, p.9]. Um componente realiza uma determinada interface fornecida se contém uma implementação de todas as operações definidas por aquela interface. Um componente possui uma interface requerida se utiliza uma operação definida naquela interface. Componentes se conectam por meio da interface requerida de um com a interface fornecida de outro [Barroca et al., 2005, p.6], conforme ilustrado na Figura A.3. Se um componente for totalmente autocontido, ele não possui interface requerida. Um componente pode requerer mais de uma interface e estar em conformidade com mais de uma interface fornecida. As dependências de contexto são definidas pelas interfaces requeridas [Szyperski, 1997, p.369].
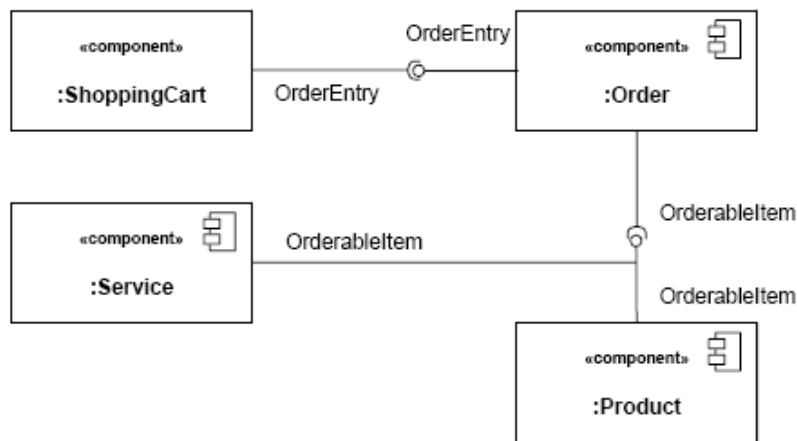
Figura A.3. Conexão entre os componentes [OMG, 2005]

A implementação de interface da orientação a objetos pode ser utilizada para implementar o conceito de interface de componente. Propriedade e eventos são mapeados na forma de métodos, porém, alguns aspectos da interface, como os aspectos não-funcionais e as interfaces requeridas, não são passíveis de representação [Gimenes & Huzita, 2005; D'Souza & Wills, 1998, p.388]. Alguns modelos de componentes definem maneiras próprias para representar a interface, como por exemplo, a IDL (*Interface Definition Language*) do CORBA [Siegel, 2000].

Um componente implementa diretamente uma interface ou implementa objetos que provêem interfaces [Szyperski, 1997, p.41]. Estes objetos são passados de componente em componente, de modo que um componente não tem ciência da origem do objeto e do código sendo executado [Szyperski, 1997, p.42]. Chama-se de **instância de componente**, o conjunto de objetos pelos quais se manipula o componente [Szyperski, 1997, p.370]. Estes objetos são a manifestação do componente em tempo de execução [D'Souza & Wills, 1998, p.390]. Código adicional, conhecido como **adaptador**, pode ser inserido entre componentes para realizar conversões simples de modo a compatibilizar interfaces nos casos em que a substituição de tipos não é suficiente.

Para utilizar um componente é necessário implantá-lo (**deployment**) em uma infra-estrutura de execução. A implantação não pressupõe a modificação do componente, entretanto, oferece a possibilidade de customização externa [Heineman, 2000; D'Souza & Wills, 1998, p.395]. A customização consiste na adaptação de um componente antes de sua instalação ou uso, normalmente com o

objetivo de especializar seu comportamento [Weinreich & Sametinger, 2001, p.42; D'Souza & Wills, 1998, p.xvii]. Como normalmente componentes são desenvolvidos no estilo *blackbox*, revelando o mínimo possível de sua implementação, as maneiras mais comuns de customizar um componente é através da modificação de propriedades ou da composição com outros componentes que especializam determinados comportamentos [Weinreich & Sametinger, 2001, p.42].

Na customização por composição, um componente contém uma referência a outro e repassa a ele as chamadas das operações. Um componente só depende da interface do outro, o que caracteriza um reuso *blackbox* [Szyperski, 1997, p.137]. No reuso *blackbox*, nenhum detalhe, além dos providos pelas interfaces do componente e por sua especificação, é disponibilizado aos clientes. Outra maneira de customizar o comportamento de um componente é modificando suas propriedades. Técnicas para isto incluem passagem de parâmetros para seus métodos, tabelas lidas pelo componente, configuração e opções na implantação. Uma abordagem comum é associar um arquivo descritor a um componente.

O arquivo descritor possibilita que o componente seja configurado sem que seja necessário recompilá-lo [Szyperski, 1997, p.33]. O arquivo descritor provê informações sobre o conteúdo do pacote, sobre as dependências externas e sobre as configurações do componente. Esta descrição é analisada pela infra-estrutura de execução e é utilizada para instalar e configurar o componente [Weinreich & Sametinger, 2001, p.44].

Componentes são *blackbox* ou *whitebox*, com as classificações intermediárias de *graybox* ou *glassbox* [Szyperski, 1997, p.29]. No reuso *whitebox*, detalhes do código interno são liberados, possibilitando, por exemplo, o uso da herança para customizar o comportamento do componente [Szyperski, 1997, p.33]. Apesar de não ser recomendada, a herança que ultrapassa as fronteiras do componente às vezes é utilizada [Szyperski, 1997, p.32]. Entretanto, esta herança cria uma dependência e acoplamento direto entre as implementações das classes, o que vai de encontro às características de componentes. D'Souza & Wills [1998, p.475] propõe uma maneira de transformar heranças em composições de objetos.

A.4.
Representação de Componentes

A documentação favorece o reuso [Sametinger, 1997]. A documentação deve ser suficiente para recuperar um componente, avaliar sua adequabilidade ao contexto do reuso, fazer adaptações e integrá-lo ao seu novo ambiente [Gimenes & Huzita, 2005]. É necessária uma notação comum entre os envolvidos para documentar e debater sobre o projeto do sistema, reduzindo a chance de má interpretação. A UML (*Unified Modeling Language*) é a linguagem padrão para representar o projeto de sistemas orientados a objetos.

O conceito de componente adotado inicialmente na UML, até a sua versão 1.5, é excessivamente abrangente se comparado com a concepção da comunidade de desenvolvimento baseado em componentes [Houston & Norris, 2001, p.257]. De acordo com os autores originais da UML: “Os componentes são empregados para a modelagem de coisas físicas que podem residir em um nó, como executáveis, bibliotecas, tabelas, arquivos e documentos” [Booch, Rumbaugh & Jacobson, 2000, p.341]. A representação de componente na UML 1.x é apresentada na Figura A.4 à esquerda.

Figura A.4. Representação de componente na UML 1.x e 2.0

A versão 2.0 da UML [OMG, 2005], cuja versão final do documento de especificação de superestrutura foi lançada em agosto de 2005, altera profundamente a concepção de componente de software em relação à versão anterior. Nesta nova concepção, um componente é visto como uma unidade modular com interfaces bem definidas e substituível no contexto do sistema. A Figura A.4 à direita apresenta o novo símbolo de componente. A UML 2.0 prevê a representação de portas, conectores, interfaces providas e requeridas. A Figura A.5 à esquerda apresenta um componente com duas interfaces providas e três requeridas, representadas na forma icônica. A Figura A.5 à direita representa as interfaces na forma expandida e a ligação entre o componente e a interface

provida através de um relacionamento de realização e a ligação com uma interface requerida através de um relacionamento de dependência.
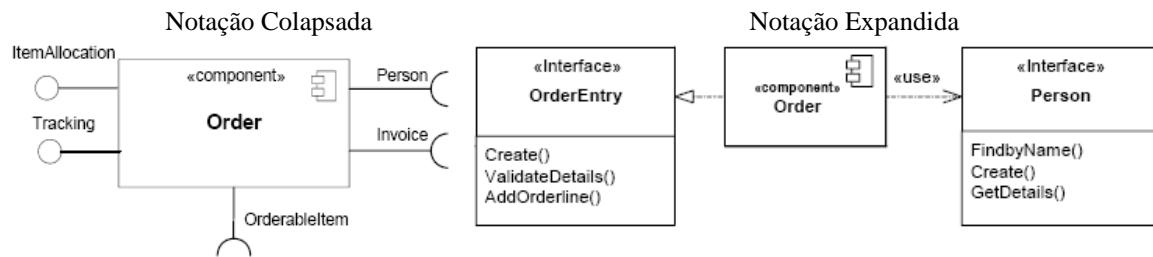
Figura A.5. Representação de interfaces na forma colapsada e expandida [OMG, 2005]

Para representar as classes que o componente implementa, é utilizado o relacionamento de dependência ou a estrutura de classes é aninhada dentro símbolo de componentes, conforme ilustrado na Figura A.6.
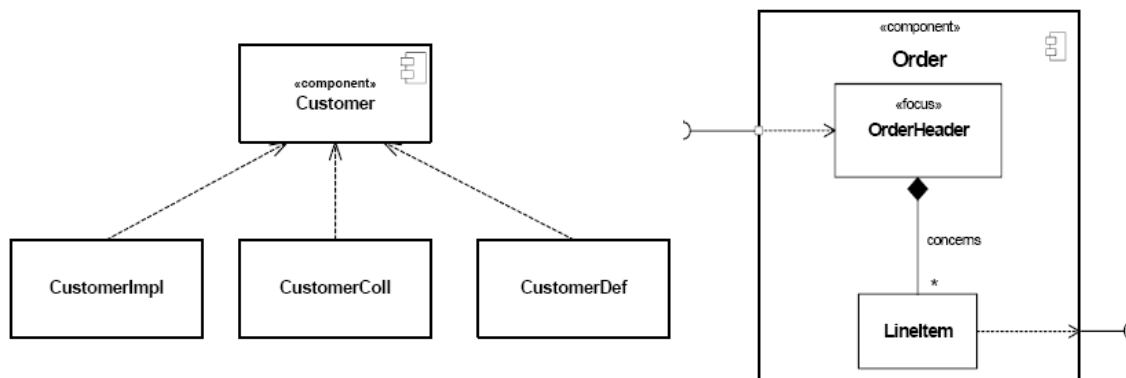
Figura A.6. Representação das classes que o componente implementa [OMG, 2005]

D’Souza & Wills [1998, p.84] propõe uma extensão à UML para representar a interface de um componente e seu modelo de objetos. Conforme ilustrado na Figura A.7, o compartimento do meio da representação da interface é utilizado para explicitar o modelo de objetos manipulado pelas operações da interface, de modo a facilitar o entendimento e o uso do componente.
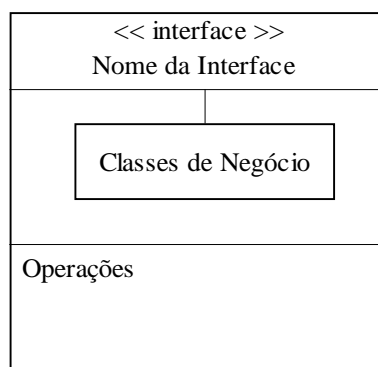
Figura A.7. Representação de um componente e seu modelo de objetos

Além da documentação gráfica da estrutura e dos inter-relacionamentos entre os componentes, é necessário descrevê-los textualmente. Nesta descrição é necessário englobar aspectos referentes aos requisitos funcionais e não-funcionais. Memória utilizada e tempo de resposta são exemplos de aspectos que precisam ser documentados, pois podem criar dependências não previstas, em um fenômeno chamado de vazamento de propriedade [Gimenes & Huzita, 2005]. Não há uma notação padrão para a descrição dos componentes. Paludo & Burnett [2005] propõem a utilização da estrutura de documentação de padrões de projeto para documentar componentes. São documentados, por exemplo, nome, intenção, aplicabilidade, variabilidade, estrutura, código de exemplo, usos conhecidos e componentes relacionados.

A.5. Implementação de Componentes

Há componentes de software implementados em linguagens procedurais, como Pascal e C, embora a maior parte dos componentes seja desenvolvida utilizando metodologias e linguagens orientadas a objetos [Vicenzi et al., 2005, p. 235; Apperly, 2001, p.29]. Para um componente o relevante é sua interface externa e não a maneira como foi implementado.

As primeiras APIs para componentes disponibilizavam apenas um conjunto de funções que componentes externos poderiam invocar, enxergando o componente com um bloco único. Atualmente, as APIs possibilitam acessar os objetos que compõe o modelo do componente [D'Souza & Wills, 1998, p.394]. Desta maneira, as interfaces que o componente disponibiliza ou requer definem o **modelo de objetos** que é compatível com aquele componente. Referências a objetos criados em um componente deixam as fronteiras e se tornam visíveis aos clientes do componente [Szyperski, 1997, p.31]. Por exemplo, é possível acessar o objeto célula do componente Planilha Eletrônica, o objeto ponto do Gerenciador Gráfico, entre outros. [D'Souza & Wills, 1998, p.394].

Outro fator importante a ser considerado na construção de componentes é o empacotamento, que possibilita que o componente seja instalado como uma unidade [Szyperski, 1997, p.276]. O empacotamento pode conter arquivos,

módulos, código executável, código fonte, código de validação, especificações, testes, documentações, etc. [D'Souza & Wills, 1998, p.386]. Os mecanismos para o empacotamento diferem de tecnologia para tecnologia [D'Souza & Wills, 1998, p.387]. Por exemplo, componentes Java são empacotados em arquivos JAR, que incluem as classes que implementam os serviços dos componentes. [D'Souza & Wills, 1998, p.400]. Os diversos fatores que definem as possibilidades de implementação de um componente são definidos no *component model* correspondente.

A.5.1. Modelo de Componentes

Ao comprar um eletrodoméstico, ele é conectável à tomada de uma casa porque a tomada, o plug e a energia disponíveis são aderentes a uma padronização. Para obter o reuso e substitutibilidade de componentes de software, eles também devem ser aderentes a um padrão, que na literatura é chamado de *modelo de componentes* (*component model*[35]). Assim como há mais de um padrão para tomada, há mais de um modelo para componentes de software. Um programador pode criar seu próprio modelo, eventualmente sendo uma especialização de um modelo disponível. Para compatibilizar componentes de um modelo para outro, adaptadores podem ser desenvolvidos.

Um modelo de componentes define vários aspectos da construção e da interação dos componentes, abordando, entre outros fatores: como especificar o componente, como instanciar o componente, quais os tipos de conectores e portas disponíveis, qual o modelo de dados padrão, como as ligações entre os objetos pertencentes a componentes diferentes são realizadas, como são feitas transações distribuídas, quais são os tipos de interface, as interfaces obrigatórias, como são tratados o catálogo e a localização dos componentes, o despacho das requisições e respostas, a segurança, o repositório, o formato, a nomeação, os meta-dados, a interoperabilidade, a documentação, os mecanismos de customização, a composição, a evolução, o controle de versões, a instalação e a desinstalação, os

[35] O termo *component model* é equivalente ao termo *component architecture* definido por D'Souza & Wills [1998].

serviços de execução, o empacotamento, etc. [Councill & Heineman, 2001, p.7, p.11; Weinreich & Sametinger, 2001, p.37; Szyperski, 1997, p.32; D'Souza & Wills, 1998, p.396, p.411].

O modelo de componentes define o padrão de descrição de interface [Weinreich & Sametinger, 2001, p.39]. Alguns modelos de componentes utilizam uma *interface description language* (IDL) independente da linguagem de programação, enquanto outros utilizam conceitos da própria linguagem ou da tecnologia para definir a interface, como é o caso das interfaces OO. Dependendo da IDL, interfaces podem incluir as exceções que podem ser lançadas, pré-condições e pós-condições para cada operação [Weinreich & Sametinger, 2001, p.39].

Um componente deve ser unicamente nomeado [Weinreich & Sametinger, 2001, p.40]. Alguns modelos de componentes definem a existências de identificadores únicos, como é o caso do COM Component Model. Outros modelos utilizam um espaço de nomes hierárquico, como é o caso das tecnologias da Sun Microsystems, que utilizam o nome do domínio invertido.

Um modelo de componentes deve definir quais são os padrões de composição. Os tipos de acoplamento mais comuns entre dois componentes são cliente/servidor e publicador/ouvinte [Weinreich & Sametinger, 2001, p.43]. No primeiro, o cliente explicitamente chama operações do servidor e, no segundo, o ouvinte se registra como tratador de eventos e informações disponibilizadas pelo publicador. O modelo de componentes define também os tipos de conectores disponíveis, que podem ser herdados da linguagem de programação correspondente ou serem próprios da tecnologia, como no caso do SOAP e IIOP.

Eventualmente, versões antigas e atuais de um componente co-existem no mesmo sistema. As regras e padrões para a evolução dos componentes e seu versionamento também fazem parte do modelo de componentes [Weinreich & Sametinger, 2001, p.44]. Um modelo de componentes também descreve como os componentes são empacotados e de que forma que eles podem ser individualmente instalados no sistema.

O modelo de componentes também define o padrão de *deployment*, que especifica a estrutura e a semântica para os arquivos descritores. O modelo de

componentes define a maneira como é feito o deployment, incluindo o eventual registro do componente e da interface [Weinreich & Sametinger, 2001, p.45]. Tipicamente o *deployment* envolve três passos: instalação do componente; configuração do componente e do ambiente de execução; e instanciação do componente para uso [Councill & Heineman, 2001, p.9].

Os modelos de componentes genéricos não provêem um modelo de negócio específico para um domínio [Weinreich & Sametinger, 2001, p.37]. Um projeto define seu próprio modelo de componentes independentemente ou como uma especialização de um existente [D'Souza & Wills, 1998, p.411]. Por exemplo, a OMG (*Object Management Group*), prevê a utilização do CORBA como base para a construção de modelos de componentes especializados. CORBAservices define a padronização voltada para serviços gerais de sistemas distribuídos, enquanto CORBAfacilities define a padronização dos serviços horizontais (comuns a vários domínios) [Weinreich & Sametinger, 2001, p.46]. Além da padronização de serviços de infra-estrutura, alguns modelos de componentes definem também uma padronização vertical, estabelecendo o modelo de objetos que os componentes interoperantes compartilham e manipulam [D'Souza & Wills, 1998, p.396].

Corba Component Model (CCM), Microsoft OLE, (D)COM/COM+, Sun EJB, JavaBeans e Web Service são alguns exemplos de modelos de componentes. Alguns modelos, como CORBA e Web Services, possibilitam o reuso de componentes que não foram necessariamente desenvolvidos na mesma linguagem de programação [Gimenes & Huzita, 2005]. Na subseção seguinte, são apresentados alguns modelos de componentes a título de exemplo.

A.5.2. Exemplos de Modelos de Componentes

O **OLE** (*Object Linking and Embedding*) é um modelo de componentes proposto pela Microsoft cujo propósito inicial era integrar em um documento objetos gerados por diversas aplicações. Ao instalar no sistema operacional aplicações compatíveis com o modelo, elas tornam-se disponíveis para receber e oferecer conteúdos. O editor de documento passa a ser um container que carrega

conteúdos disponibilizados pelas demais aplicações, como gráficos, sons, vídeo, figuras, conforme ilustrado na Figura A.8. Ao instalar uma nova aplicação, não é necessário modificar as demais para que elas troquem conteúdos.
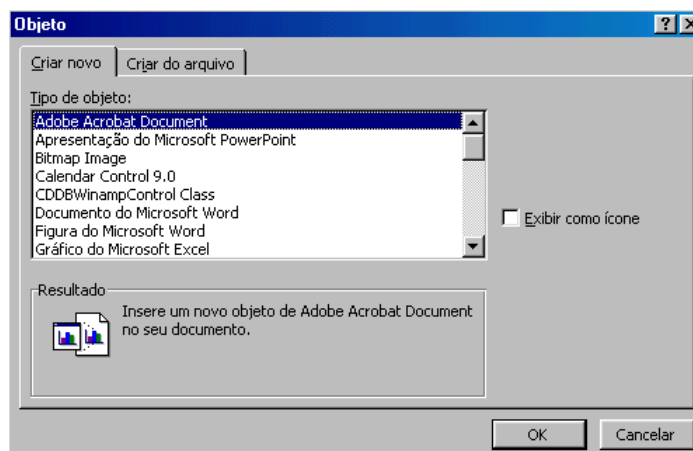
Figura A.8. Inserção de um objeto OLE em um documento do Microsoft Word

COM, **DCOM** e **ActiveX** são outros modelos de componentes utilizados pela Microsoft que vieram da evolução das idéias do modelo original OLE [Brockschmidt, 1996]. Os modelos oferecem suporte a diversos níveis de complexidade de componentes e possibilitam a integração de aplicações desenvolvidas por diferentes fabricantes. A solução enfoca componentes binários interoperáveis.

A Microsoft definiu um padrão para seus componentes de interface com o usuário (widgets), disponibilizados através da linguagem Visual Basic. O modelo introduzido foi o **VBX**, que posteriormente foi substituído pelo **OCX**, pelo ActiveX e depois pelo COM. Atualmente, o modelo de componentes do **.NET** unifica toda a tecnologia de componentes da Microsoft. A Borland desenvolveu seu próprio modelo de componentes de interface, o **CLX** (*Component Library for Cross Platform*), que possibilita o desenvolvimento de aplicações para o Microsoft Windows e para o Linux através dos ambientes Kylix, Delphi e C++ Builder.

O **CORBA** (*Common Object Request Broker Architecture*) fornece um modelo de componentes que possibilita a comunicação entre componentes distribuídos. O CORBA utiliza a IDL (*Interface Definition Language*) para descrever a interface pública de seus serviços. O cliente não é dependente da localização do objeto que está sendo invocado, da linguagem que o mesmo foi programado ou do sistema operacional que está sendo utilizado.

Web Services é um padrão para comunicação entre componentes através da web. A aplicação utiliza os serviços dos componentes através do protocolo SOAP, que encapsula as chamadas dos serviços e os retornos em pacotes XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
http://java.sun.com/xml/ns/portlet" xmlns="http://java.sun.com/xml/ns/portlet">
        <portlet>
              <portlet-name>helloWorld</portlet-name>
                <portlet-class>examples.helloworld.HelloWorld</portlet-class>
              <portlet-info>
                        <title>Hello World</title>
                </portlet-info>
        </portlet>
</portlet-app>
```

Figura A.9. Arquivo descritor de um portlet

JavaBeans é o modelo básico de componentes da Sun, a partir do qual são feitas diversas extensões. O JavaBeans define como tratar no componente eventos, propriedades, introspecção, reflexão, customização e empacotamento [Szyperski, 1997]. **Enterprise Java Beans** é o modelo para a interconexão de componentes remotos em aplicações corporativas. Recentemente a Sun lançou o modelo de componentes para interface com o usuário **Java Server Faces** (JSF) e o modelo **Portlets** (JSR 168) para o reuso de componentes em portais na web. Utilizando este modelo é possível reusar um componente desenvolvido para um portal em outro, de modo que uma instituição monta seu portal buscando ou adquirindo componentes de terceiros. Cada componente possui um arquivo descritor (Figura A.9) que define suas configurações e implementa uma interface que define os métodos ativados durante seu ciclo de vida (Figura A.10).

```
package examples.helloworld;

 import java.io.*;
 import javax.portlet.*;

 public class HelloWorld extends GenericPortlet
 {
 public void render(RenderRequest request, RenderResponse response)
 throws PortletException, IOException
     {
             response.getWriter().write("<p>Hello World</p>");
     }
 }
```

Figura A.10. Implementação de um método referente ao ciclo de vida de um portlet

Para executar estes componentes, é necessária uma infra-estrutura de execução específica, que cuida do ciclo de vida, agregação, segurança, configuração, persistência, etc. Por exemplo, no caso do Portlets, o container aciona os componentes que geram dinamicamente o conteúdo exibido para o usuário, agregando a saída de cada um em uma única apresentação.

A.6. Infra-estrutura de Execução

Um container é uma plataforma, normalmente desenvolvida por terceiros, com o objetivo de hospedar e gerenciar componentes de um determinado modelo, provendo a eles serviços de infra-estrutura de execução. O acesso remoto, o gerenciamento de transações distribuídas, o pooling de recursos, o acesso concorrente, o clustering, a tolerância a falhas, a autenticação, a persistência, a configuração, o gerenciamento de permissões e de sessão são exemplos de serviços providos por containeres [D'Souza & Wills, 1998, p.401].

O container libera o desenvolvedor de implementar serviços técnicos de sistema de baixo nível, direcionando seus esforços para as regras de negócio e para a composição do sistema. Em alguns casos, o uso de container também possibilita alterar aspectos não relacionados com a lógica do negócio sem ter que alterar a aplicação, bastando re-configurar o container para mudar aspectos como segurança, permissões, logging, banco de dados, etc. A máquina virtual Java é um container para executar componentes Java (arquivos .class ou .jar). Algumas arquiteturas estendem o container para gerenciar componentes mais especializados como servlets ou applets [D'Souza & Wills, 1998, p.460]. Tomcat, JBoss, Webspheare, Pluto e JetSpeed são exemplos de containeres.

Component framework[36] é um conjunto de elementos de software, regras e contratos que governam a execução de componentes que estão em conformidade com um modelo de componentes [Szyperski, 1997, p.369]. O *component framework* define as invariantes e os protocolos de conexão entre os componentes, estabelece as "condições ambientais" para as instâncias dos componentes e regula as interações entre elas [Szyperski, 1997, p.276]. O *component framework* define uma interface chamada de *lifecycle interface* que estabelece a conexão com os componentes [Schwarz et al., 2003]. Esta interface é acessada para gerenciar,

[36] *Component framework* é um termo com diversos entendimentos na literatura. Neste trabalho, está sendo seguida a definição proposta por Szyperski [1997]. *Component framework* é equivalente ao conceito de *component model implementation* definido por Councill & Heineman [2001, p.7].

inicializar, executar e desativar os componentes. O *component framework* não acessa as interfaces de negócio dos componentes.

O *component framework* segue e oferece suporte a um modelo de componentes provendo uma infra-estrutura para apoiar sua execução. O *component framework* fornece uma base para a integração dinâmica dos componentes, até de diferentes fabricantes, de acordo com as necessidades e preferências dos usuários e desenvolvedores. O *component framework* oferece serviços de execução como criação de objetos, gerenciamento de ciclo de vida, persistência de objetos, licenciamento, acesso a dados, gerenciamento de transações, balanceamento de carga, etc. *Component frameworks* para sistemas distribuídos oferecem serviços adicionais, como modos de conexão e comunicação remota, notificação de eventos, localização de serviços e segurança [Weinreich & Sametinger, 2001, p.45]. Eventualmente, a interface ou o componente precisa ser registrado no *component framework* antes de ser utilizado [Councill & Heineman, 2001, p.12].

A relação entre um componente e um *component framework* é similar à relação entre um programa e o sistema operacional. Um sistema operacional provê um ambiente de execução para aplicativos, colocando uma camada sobre o hardware, e provendo serviços de baixo nível, como acesso a dispositivos de E/S, gerenciamento de memória, etc. [Weinreich & Sametinger, 2001, p.34]. O sistema operacional regula o acesso dos aplicativos aos recursos, oferece serviços de infra-estrutura, define os mecanismos de comunicação e interoperabilidade entre os componentes e oferece suporte à instalação e registro dos componentes (aplicativos). Os aplicativos são utilizados para montar o ambiente de trabalho e o sistema operacional define parcialmente a arquitetura do sistema [Szyperski, 1997, p.274].

O conceito de *component framework* está ligado à idéia de linha de produto [Barroca et al., 2005, p.20]. Uma linha de produtos de software é um conjunto de produtos que compartilham um conjunto de características visando satisfazer um determinado segmento de mercado ou uma missão específica [Clements & Northrop, 2001]. Para reduzir os custos de desenvolvimento e manutenção, os produtos são gerados de uma maneira sistemática a partir de um núcleo de artefatos.

A.7. Component Kits

Um *component kit* é uma coleção de componentes que foram projetados para trabalhar em conjunto [D'Souza & Wills, 1998, p.404]. *Component kits* são freqüentemente utilizados para construir interfaces gráficas para aplicações a partir de botões, formulários, listas, etc. De um kit de componentes gera-se uma família de soluções, fazendo diferentes arranjos e eventualmente desenvolvendo alguns sob medida [Wills, 2001, p.309]. D'Souza & Wills [1998] ilustram a utilização de *component kits* na eletrônica. A partir de um conjunto de componentes interoperáveis e padronizados, são construídos circuitos para os mais variados propósitos.

A Figura A.11 ilustra a geração de um kit de componentes a partir de um conjunto de aplicações similares e a posterior construção de novas aplicações a partir deste kit. Inicialmente, o desenvolvedor analisa aplicações similares, identificando e generalizando componentes comuns, criando um kit de componentes. Tendo o kit, os componentes são utilizados para montar novas aplicações da mesma família [D'Souza & Wills, 1998, p.385].
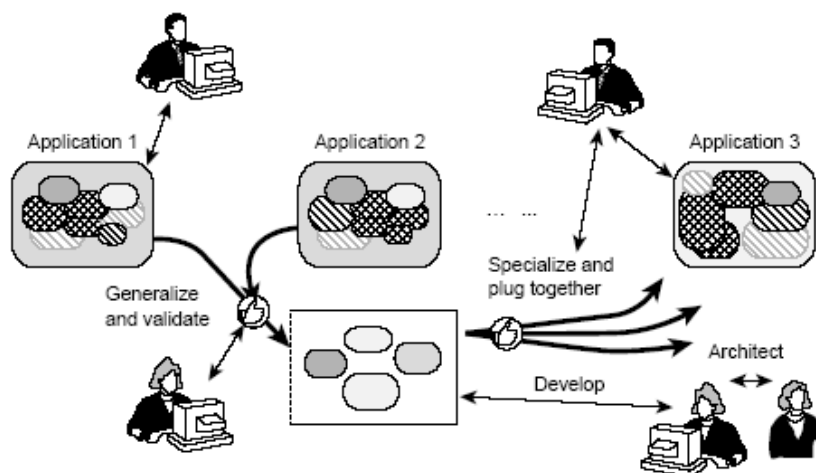
Figura A.11. Desenvolvimento de um kit de componente e construção de aplicações a partir dele [D'Souza & Wills, 1998, p.385]

Um *component kit* é projetado para um modelo de componentes específico, o que favorece a interoperabilidade [D'Souza & Wills, 1998, p.428]. Um kit não necessariamente possui um conjunto fixo de componentes, podendo ser adicionado novos, respeitando a arquitetura definida. As diversas aplicações

compartilham os componentes, que por sua vez executam na infra-estrutura de execução.

Um toolkit é um tipo de SDK (*Software Development Kit*), que apresenta, além dos componentes, um conjunto de ferramentas para criar aplicações para uma determinada plataforma, sistema ou framework. Os toolkits são mais comumente encontrados para componentes de interface com o usuário, também chamados de widgets. Alguns exemplos de toolkits de widgets para a linguagem Java são o AWT (*Abstract Windowing Toolkit*), Swing e SWT (*Standard Widget Toolkit*).

Toolkits possibilitam que programadores desenvolvam software a partir de componentes prontos, mesmo sem muita experiência de programação, no estilo das ferramentas RAD (*Rapid Application Development*). Os toolkits favorecem a criatividade dos desenvolvedores, o que é fundamental para áreas não bem estabelecidas. Ao encapsular detalhes de implementação de baixo nível, os toolkits liberam os desenvolvedores para pensar em novas interfaces e mecanismos de interação, e possibilitam uma prototipação rápida para testar, refinar e validar as idéias [Greenberg, 2006].

A.8. Arquitetura de Software Baseada em Componentes

Uma arquitetura define a estrutura[37] de um software, descrevendo as propriedades, restrições e relacionamentos de suas partes [Stafford & Wolf, 2001, p.373]. Ela representa um conjunto de restrições e regras, definindo os elementos, conectores, protocolos, componentes, propriedades visíveis e a interação e a função de cada parte no contexto do sistema [D'Souza & Wills, 1998, p.481; Bass et al., 2003]. A arquitetura é uma representação abstrata de alto nível do projeto de um software. A granularidade dos componentes da arquitetura varia de pequenos pedaços de código a aplicações inteiras, como SGBDs ou servidores de e-mail. As

---

[37] Na literatura, muitas vezes o termo arquitetura também é utilizado para representar a infra-estrutura da aplicação, em frases do tipo "o componente interage com a arquitetura da aplicação".

conexões entre os componentes abstraem como eles de fato interagem, como por exemplo, chamada de método, compartilhamento de dados, pipes, RPCs (*Remote Procedure Calls*), sockets, etc. [D'Souza & Wills, 1998].

Um estilo arquitetural descreve uma família de arquiteturas de software que compartilham propriedades, como por exemplo, os componentes permitidos, as restrições e interações entre os componentes, as invariantes, o modelo computacional, etc. [Stafford & Wolf, 2001, p.383]. Fluxo de dados, máquina virtual, chamada de procedimento, MVC (*Model View Controller*), cliente-servidor, peer-to-peer, blackboard, camadas e orientação a serviços são exemplos de estilos arquiteturais [Barroca et al., 2005, p.17]. Cada estilo normalmente endereça um aspecto específico do sistema, sendo portanto possível de utilizar mais de um estilo na mesma arquitetura. Além disto, cada componente de um sistema pode ter uma arquitetura e estilo arquitetural próprios, desde que a parte externa do componente seja compatível com a arquitetura da aplicação.

Apesar da arquitetura ser única, pode-se fornecer várias visões sobre ela [D'Souza & Wills, 1998, p.483]. Cada visão enfoca um determinado aspecto da arquitetura, omitindo os demais [Stafford & Wolf, 2001, p.386]. Algumas visões são mais apropriadas para o desenvolvimento do sistema, outras para o reuso, outras para o teste e implantação.

O desenvolvimento baseado em componentes considera pelo menos duas visões da arquitetura: arquitetura de aplicação e arquitetura técnica [D'Souza & Wills, 1998, p.483]. Na arquitetura de aplicação, a preocupação é com a estrutura dos componentes do domínio, representando um projeto lógico de alto nível independente da tecnologia de suporte. Nesta arquitetura, são mostradas a função de cada componente no contexto do sistema e a interação entre eles. A arquitetura de aplicação consiste de um conjunto de decisões sobre a plataforma, um conjunto de *component frameworks* e os mecanismos de interoperação entre eles [Szyperski, 1997, p.275]. A arquitetura técnica considera os detalhes da tecnologia de componentes a ser utilizada e é totalmente independente do domínio da aplicação. A arquitetura técnica retrata as tecnologias de comunicação entre os componentes (TCP/IP, ODBC, etc.) e aspectos referentes a escalabilidade e performance.

A.9. Frameworks

O conceito de frameworks está relacionado ao de componentes; são conceitos complementares que contribuem para o reuso de software [Gimenes & Huzita, 2005]. Um framework é uma infra-estrutura reusável de todo ou de parte de um sistema, com o objetivo de ser instanciado para resolver uma família de problemas [Govoni, 1999]. As partes invariantes de um domínio são implementadas no framework e reusadas nas instanciações. O framework oferece uma estrutura comum para um domínio de aplicações, promovendo o reuso do conteúdo conceitual do domínio de um software ou da solução de um problema [Gimenes & Huzita, 2005].

Ao definir uma arquitetura parcialmente implementada e encapsular detalhes de implementação, o framework libera os desenvolvedores de terem que entender as complexidades envolvidas com a solução do problema e com o domínio utilizado [Pree, 1995]. O framework é normalmente construído por especialistas em um domínio particular e utilizado por leigos naquele domínio [Lajoie & Keller, 1995]. O reuso proporcionado pelo uso de um framework não atinge somente a implementação, se refletindo também na análise e no projeto do sistema.

Alguns frameworks são voltados à solução de problemas ligados à tecnologia, como interface com o usuário, persistência de objetos e suporte a MVC. Outros frameworks são voltados para um determinado domínio, como por exemplo, aplicações bancárias e relacionamento com o cliente. Estes frameworks são embasados em teorias e modelos do domínio e definem uma arquitetura orientada para a área de aplicação específica.

A utilização de frameworks traz as vantagens associadas ao reuso de código: aumento da qualidade, redução do esforço de implementação, direcionamento dos esforços para os aspectos específicos da solução, etc. Contudo, conforme discutido anteriormente, o reuso, principalmente de código proveniente de terceiros, pode trazer complicações adicionais ao desenvolvimento. No caso de

framework, como muitas vezes ele assume o fluxo da aplicação[38], para alterar os processos de execução pode ser necessário alterar o código do framework.

A construção de um framework não é simples, e deve ser planejada para o reuso [Mattsson, 2000]. O framework deve ser: flexível (reusar as abstrações em diversos contextos), extensível (permitir a adição ou modificação de funcionalidades) e compreensível (bem documentado, seguindo padrões e provendo exemplos de utilização) [Gimenes & Huzita, 2005]. Para desenvolver um framework, deve-se identificar e caracterizar o domínio do problema e definir a arquitetura, o projeto da solução e as maneiras de interagir e estender o framework. Como é muito difícil prever em um primeiro momento todas variabilidades e requisitos de um framework, seu desenvolvimento normalmente é feito iterativamente.

Um framework normalmente é implementado orientado a objetos, composto de classes abstratas e concretas, interfaces e arquivos de configuração. Para especializar o framework utiliza-se herança, composição ou configuração. Os pontos de extensão do framework são chamados *hot-spots* ou *plug points* [Johnson, 1997; Govoni, 1999]. O reuso do framework, também chamado de instanciação, consiste em preencher os hot-spots para obter a aplicação final. As partes do framework que são invariantes são chamadas de frozen-spots.

A construção de um framework ou componente pode ser feita independente dos outros elementos, desde que mantida a interface e as propriedades requeridas pela arquitetura. Um framework pode ser instanciado a partir de componentes e um componente pode ser construído a partir de um framework [Oliveira, 2001]. Muitas vezes frameworks OO são utilizados para gerar uma família de componentes. Da mesma forma, frameworks complementares podem ser utilizados, tanto no nível de aplicação quanto no nível do domínio. Pode-se, por exemplo, utilizar um framework de infra-estrutura para a visão, um outro para persistência e o componente estar estruturado de acordo com um framework de domínio.

---

[38] A inversão de controle costuma ser utilizada em frameworks para reduzir o acoplamento entre a aplicação e o framework. Este princípio é chamado na literatura de princípio de Hollywood: “Não nos ligue, nós ligamos para você” [Larman, 2004]. Em vez de a aplicação chamar o framework, o framework chama a aplicação em métodos pré-definidos.

A.10. Considerações Finais

Nas linguagens de programação, o suporte ao reuso de código iniciou-se com a possibilidade do agrupamento de funcionalidade, onde linhas de código são encapsuladas e rotuladas em unidades denominadas procedimentos ou funções para serem reusadas em diversos pontos do código [Salus, 1998]. Com o tempo, passou-se a perceber a utilidade de agrupar funções relacionadas na forma de bibliotecas, para serem reusadas entre aplicações distintas. Linguagens de programação orientadas a objetos aumentaram a possibilidade de reuso através de unidades denominadas classes, que encapsulam dados e funções e possibilitam a herança de código e o polimorfismo de objetos. Da mesma maneira que o reuso de funções específicas evoluiu para bibliotecas de funções, o reuso de conjuntos relacionados de classes evoluiu para frameworks e componentes [Oliveira, 2001]. Um framework provê um conjunto genérico de classes que deve ser completado para instanciar uma aplicação específica, enquanto o uso de componentes possibilita construir um sistema compondo-o a partir de unidades de execução.

A idéia de que o software deveria ser componentizado surgiu na conferência da NATO, que lançou o termo Engenharia de Software, na Alemanha em 1968. Nesta conferência, Doug McIlroy [1968], em um artigo intitulado “Mass Produced Software Components”, levantou a necessidade de agrupar coerentemente rotinas relacionadas de modo a montar componentes reusáveis em diversos sistemas. Este reuso, além de economizar esforço de desenvolvimento, possibilitaria a produção de programas a partir de partes já testadas e amplamente utilizadas [Gimenes & Huzita, 2005]. O surgimento dos componentes para interface com o usuário, popularizado em 1992, com o Visual Basic eXtension, ou VBX, ajudou a alavancar a utilização de componentes [Oliveira, 2001]. Diversos estudos apontam a tecnologia de componentes como promissora para melhorar o reuso e a manutenibilidade de software [Szyperski, 1997, p.18].

Componentes de software são autocontidos (são relativamente independentes, fracamente acoplados e encapsulam seu projeto e implementação, incluindo dados e funcionalidades); reusáveis (podem ser utilizados por terceiros, que não necessariamente precisam ter acesso ao seu código fonte); substituíveis

(podem ser trocados por outros componentes compatíveis) e provêem serviços específicos de uma maneira coesa, bem definida e padronizada. Um componente de software pode ser desenvolvido e mantido independentemente e interage com outros componentes ou com a infra-estrutura da aplicação. A orientação a objetos é uma das melhores maneiras de implementar componentes [Szyperski, 1997, p.13; D'Souza & Wills, 1998].

A montagem de sistemas a partir de componentes está no meio termo entre configuração e programação [Morch, 1997]. É mais alto nível do que a programação, mas não tão limitada quanto a configuração. Através do uso de componentes, o sistema pode ser estendido para acompanhar a evolução dos processos de trabalho, das necessidades e da experiência com o uso do sistema. Com componentes, são reduzidas as interdependências fixas no código fonte da aplicação, melhorando a extensibilidade e facilitando a adaptação e extensão.

A componentização normalmente é disponível apenas aos desenvolvedores, que liberam diferentes versões da aplicação alterando a configuração e os componentes. Quando for necessário prover flexibilidade para o usuário final, pode-se disponibilizar a ele o conceito de componente, bem como os mecanismos para instalar e desinstalar componentes da aplicação. Estes componentes muitas vezes são do estilo "plug & play", que imediatamente se tornam disponíveis para uso após o *deployment*. Ao invés de apresentar ao usuário uma aplicação que coloca um prego ou um parafuso na parede, ele é equipado com uma caixa de ferramentas contendo, entre outras coisas, martelo, chave de fenda e parafusadeira, de onde ele escolhe a ferramenta mais adequada às suas preferências e à tarefa em questão. Adicionalmente, ferramentas existentes podem ser combinadas para formar novas ferramentas mais complexas para a realização de tarefas específicas. Entretanto, esta flexibilidade para o usuário final deve ser utilizada com cautela, pois abre espaço para "aberrações", combinando componentes incompatíveis e destoantes.

Apêndice B
Descrição dos Componentes de Colaboração

A deficiência na documentação gera um acréscimo na dificuldade de entendimento e reuso dos componentes [D'Souza & Wills, 1998]. Este apêndice contém a descrição dos componentes de colaboração, classificados em comunicação, coordenação e cooperação. A estrutura de documentação utilizada para a descrição é uma adaptação do formato utilizado para descrever padrões de projeto [Paludo & Burnett, 2005]: para cada componente são descritos nome, intenção, aplicabilidade, variabilidade, estrutura, usos conhecidos e componentes relacionados. Por enquanto, os usos conhecidos limitam-se ao ambiente AulaNet, pois até o momento é o único sistema construído de fato utilizando os componentes 3C. As interfaces dos componentes são representadas utilizando a extensão da UML proposta por D'Souza & Wills [1998], apresentada no Apêndice A.

B.1.Componentes de Comunicação

Os componentes de comunicação oferecem suporte à troca de mensagens, à negociação e à argumentação. A seguir, são descritos os componentes MessageMgr, TextualMediaMgr, DiscreteChannelMgr, MetaInformationMgr, CategorizationMgr e DialogStructureMgr.

B.1.1.MessageMgr

Nome: MessageMgr

Intenção: O componente MessageMgr oferece suporte a mensagens. As mensagens podem ser públicas, privadas, do sistema ou de controle. O componente possibilita o anexo de arquivos na mensagem.

Aplicabilidade: O componente é aplicado nas ferramentas de comunicação síncronas e assíncronas que lidam com mensagens.

Variabilidade: Tipos de mensagens.

Estrutura:

O componente oferece as interfaces IMessageMgr e IMessageMgrConfig. A primeira disponibiliza as operações relativas ao gerenciamento das mensagens. A segunda possibilita a configuração do componente e dos tipos de mensagens. O componente requer o objeto Participant, que pode ser o emissor ou receptor da mensagem, e são disponibilizados os objetos da classe Message, que representa a mensagem em si, e o objeto MessageType, que representa o tipo da mensagem. Caso a mensagem não tenha remetente, a mensagem é do sistema, e caso não tenha destinatário, é endereçada a todos participantes. A mensagem possibilita o anexo de um ou mais arquivos e disponibiliza um corpo textual (*body*), que é utilizado quando não é necessário conteúdo multimídia. Nos casos em que é necessário, o serviço que utiliza este componente deve estender a mensagem. As propriedades da classe Message são implementadas através de gets e sets. Os tipos de mensagens default são PUB (mensagem pública), PVT (mensagem privada) e CTRL (mensagem de controle). A estrutura do componente está representada na Figura B.1.
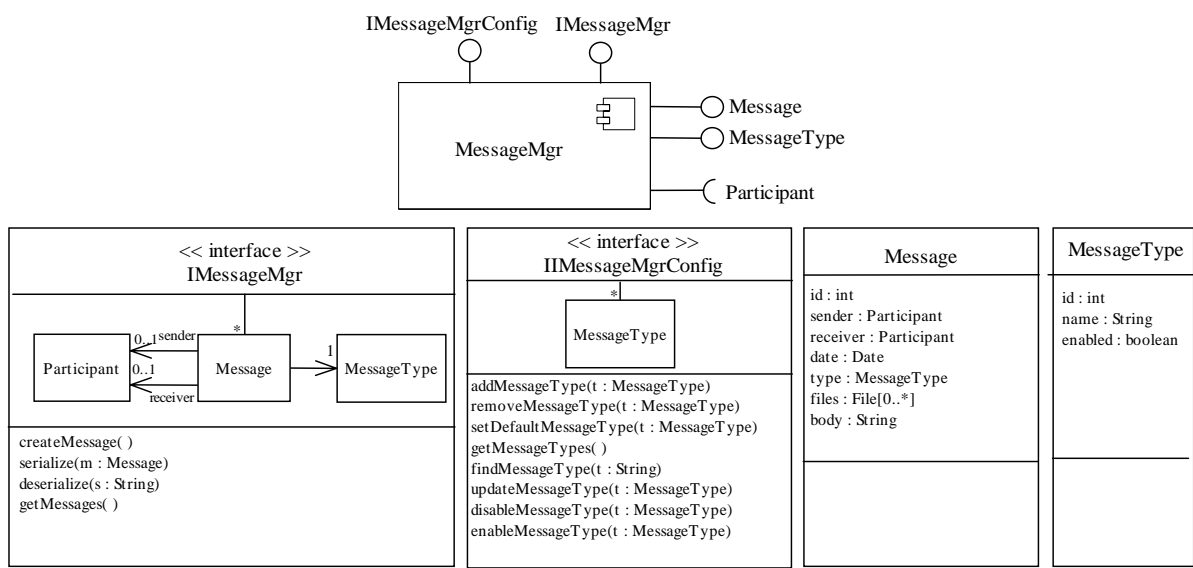
Figura B.1. Componente MessageMgr e suas interfaces

Usos conhecidos: Conferências, Debate, Correio para Turma, Correio para Participante, Mensageiro.

Componentes relacionados: TextualMediaMgr, VideoMediaMgr, AudioMediaMgr, PictorialMediaMgr e ParticipantMgr.

B.1.2.TextualMediaMgr

Nome: TextualMediaMgr

Intenção: Utilizado para mensagens com corpo textual. É possível configurar o tamanho do texto e o vocabulário disponível.

Aplicabilidade: É utilizado quando o corpo das mensagens do serviço necessita de tratamento ou codificação especial.

Variabilidade: É possível configurar o tamanho mínimo e máximo das mensagens e o vocabulário permitido ou proibido.

Estrutura:

A estrutura do componente está representada na Figura B.2. O componente oferece as interfaces ITextualMediaMgr e ITextualMediaMgrConfig, que disponibilizam serviços relativos à utilização e configuração do componente. O componente implementa a interface IMediaContent, que é o contrato geral para os conteúdos multimídia. Não é possível configurar ao mesmo tempo vocabulário permitido e proibido.
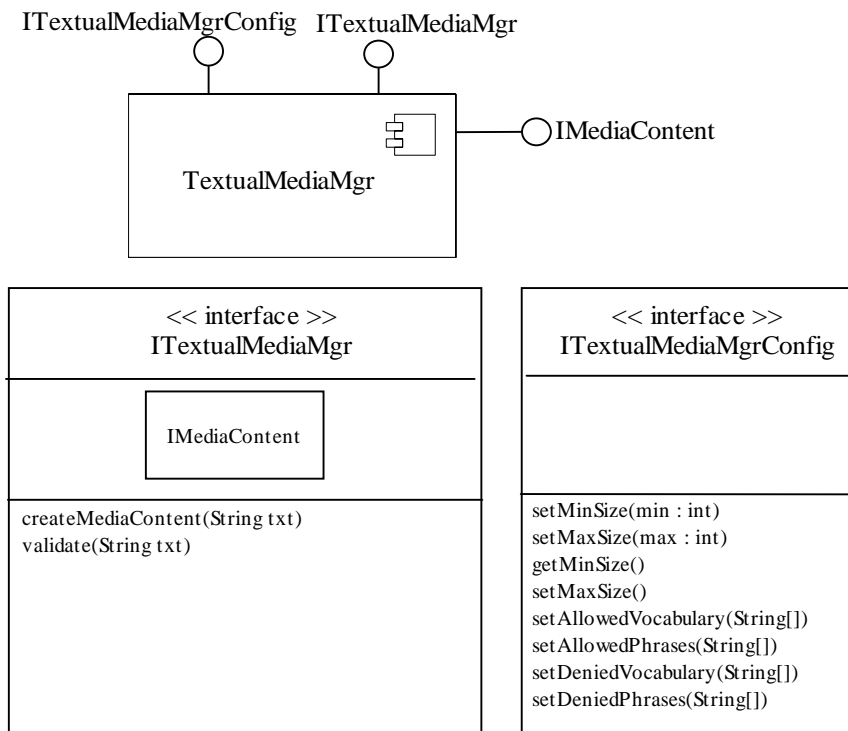
Figura B.2. Componente TextualMediaMgr e suas interfaces

Usos conhecidos: Debate

Componentes relacionados: MessageMgr

B.1.3.DiscreteChannelMgr

Nome: DiscreteChannelMgr

Intenção: Este componente é utilizado para transmitir e filtrar as mensagens. Pode ser configurado para modo síncrono ou assíncrono, que influencia a maneira como as mensagens são tratadas; *push* ou *pull*, que define como a mensagem é entregue ao participante; e o atraso na entrega, que pode ser utilizado para regular a interação em uma ferramenta síncrona [Pimentel et al., 2005].

Aplicabilidade: É utilizado quando se deseja filtrar mensagens para um determinado participante, utilizar uma interface rica (RIA) no lado cliente ou fazer *push* de mensagens (enviar sem o participante solicitar explicitamente), modo que é utilizado principalmente nas ferramentas de comunicação síncronas.

Variabilidade: Modo de entrega e atraso no envio.

Estrutura:

A estrutura do componente está representada na Figura B.3. A utilização do componente é feita pela interface IDiscreteChannelMgr, que é responsável pelo envio das mensagens para os ouvintes, que são configurados através da interface IDiscreteChannelMgrConfig.
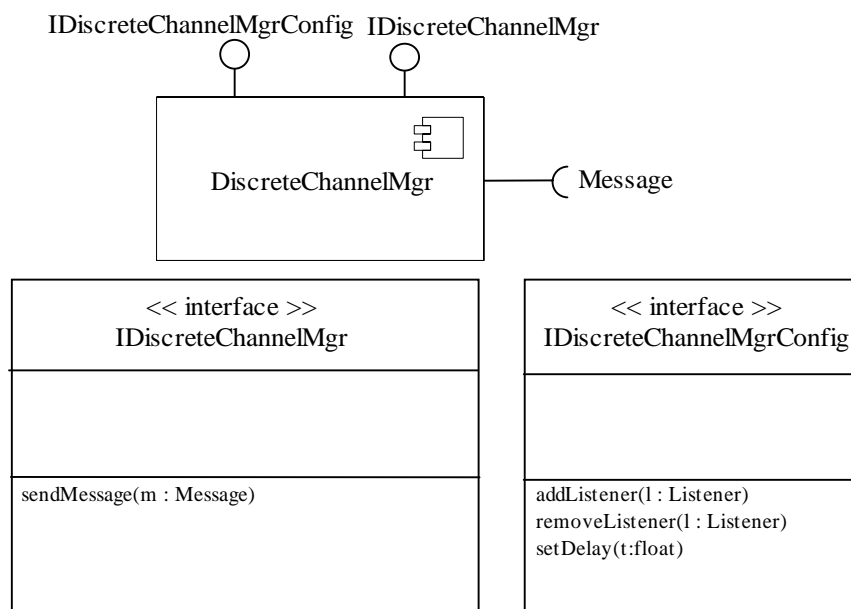
Figura B.3. Componente DiscreteChannelMgr e suas interfaces

Usos conhecidos: Debate, Correio para Participante e Mensageiro.

Componentes relacionados: MessageMgr

B.1.4.MetaInformationMgr

Nome: MetaInformationMgr

Intenção: Gerencia meta-informações associadas às mensagens.

Aplicabilidade: É utilizado quando se deseja disponibilizar meta-informações a partir das quais se deseja localizar e filtrar as mensagens.

Variabilidade: As meta-informações e sua obrigatoriedade.

Estrutura:

A estrutura do componente está representada na Figura B.4. As meta-informações são configuradas através da interface IMetaInformationMgrConfig, enquanto a atribuição das meta-informação às mensagens e a busca das

mensagens a partir das meta-informações é feita através da interface IMetaInformationMgr.
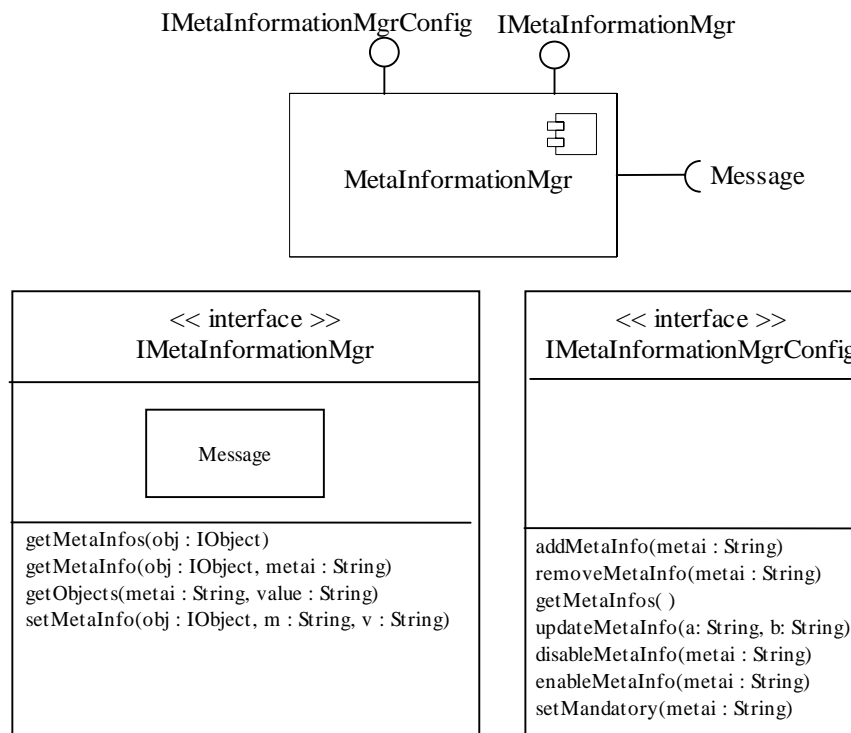
Figura B.4. Componente MetaInformationMgr e suas interfaces

Usos conhecidos: -

Componentes relacionados: MessageMgr

B.1.5.CategorizationMgr

Nome: CategorizationMgr

Intenção: Gerencia a categorização de mensagens.

Aplicabilidade: A categorização de mensagens oferece um novo elemento à linguagem da comunicação. Os participantes contextualizam a interpretação da mensagem a partir da categoria selecionada. A categorização de mensagens pode ser utilizada para complementar a estruturação do diálogo, oferecendo semântica aos inter-relacionamentos das mensagens [Gerosa et al., 2001].

Variabilidade: Pode-se configurar o conjunto de categorias e a categoria genérica.

Estrutura:

O componente CategorizationMgr possui duas interfaces fornecidas (ICategorizationMgr e ICategorizationMgrConfig) e uma interface requerida (ICategorizableObj), conforme ilustrado na Figura B.5. A interface ICategorizationMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, atribuição de categoria, recuperação dos objetos, etc. A interface ICategorizationMgrConfig disponibiliza os serviços relativos à configuração do componente, como manipulação do conjunto de categorias e seleção da categoria genérica. O componente pode ser configurado através desta interface e por seu arquivo descritor. O componente lida com objetos categorizáveis, que implementem a interface ICategorizableObj.
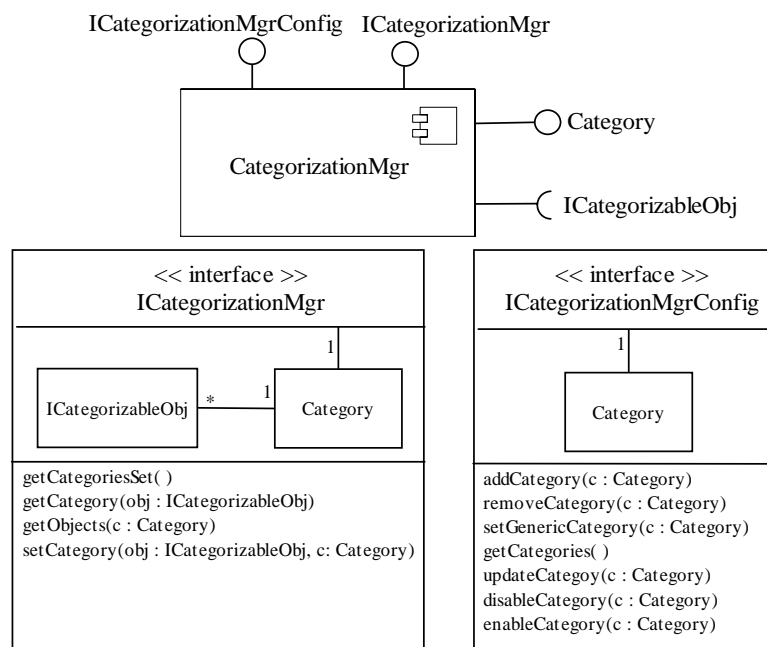
Figura B.5. Componente CategorizationMgr e suas interfaces

Usos conhecidos: Conferências e Correio para Turma.

Componentes relacionados: MessageMgr

B.1.6.DialogStructureMgr

Nome: DialogStructureMgr

Intenção: Gerencia as inter-relações entre as mensagens, que podem ser na forma de lista, árvore ou grafo.

Aplicabilidade: Este componente é utilizado principalmente nas ferramentas que adotam uma estruturação hierárquica ou em rede.

Variabilidade: O tipo de estruturação utilizado e a ordem de percurso da estrutura.

Estrutura:

A estrutura do componente está representada na Figura B.6. Através da interface IDialogStructureMgrConfig, o componente é configurado, e através da interface IDialogStructureMgr o componente é utilizado. É possível incluir as mensagens e obter os pais e filhos de uma mensagem e as raízes do diálogo.
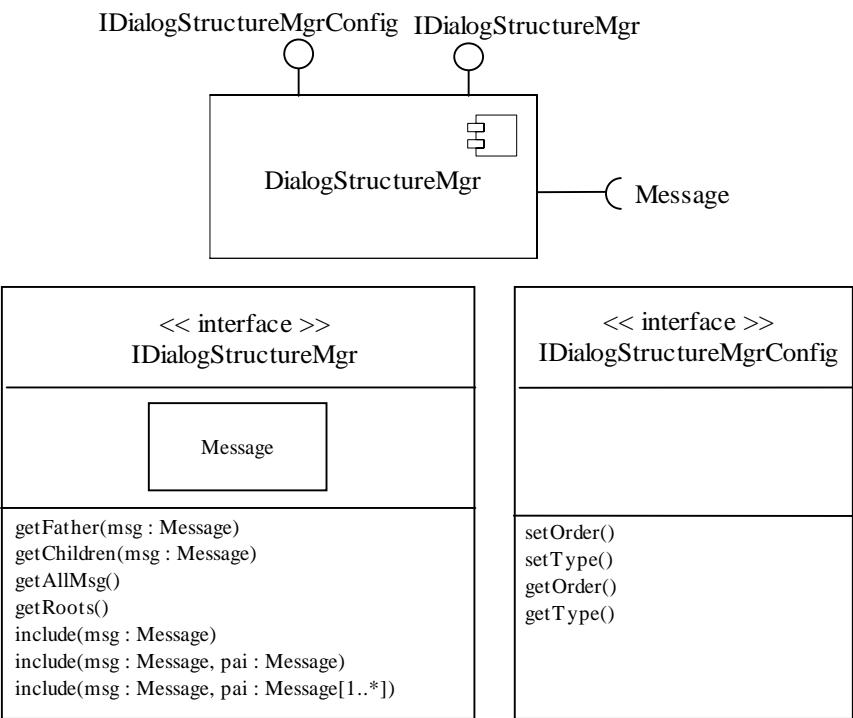
Figura B.6. Componente DialogStructureMgr e suas interfaces

Usos conhecidos: Correio para Turma, Correio para Participante e Conferências.

Componentes relacionados: MessageMgr

B.2.Componentes de Coordenação

Os componentes de coordenação oferecem suporte à organização do grupo, ao gerenciamento do acesso dos participantes e à distribuição de tarefas atribuídas. A seguir, são descritos os componentes AssessmentMgr, RoleMgr, PermissionMgr, ParticipantMgr, GroupMgr, SessionMgr, FloorControlMgr, TaskMgr, AwarenessMgr, AvailabilityMgr e NotificationMg.

B.2.1.AssessmentMgr

Nome: AssessmentMgr

Intenção: Oferece suporte à avaliação qualitativa, possibilitando a atribuição de notas aos objetos.

Aplicabilidade: A avaliação de mensagens possibilita acompanhar e incentivar a participação com qualidade nas tarefas colaborativas.

Variabilidade: Regras da avaliação, modelo de avaliação e fator de correção pela quantidade de mensagens.

Estrutura:

O componente AssessmentMgr oferece a interface para configuração do componente IAssessmentMgrConfig e para utilização IAssessmentMgr. As avaliações são referentes aos objetos de cooperação. A Figura B.7 ilustra a estrutura do componente.
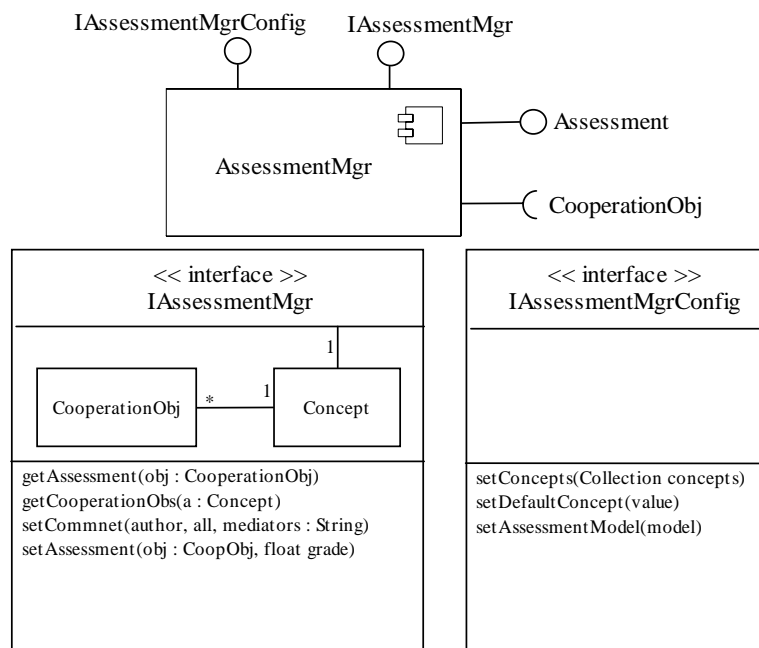
Figura B.7. Componente AssessmentMgr e suas interfaces

Usos conhecidos: Conferências, Correio para Turma e Debate.

Componentes relacionados: CooperationObjMgr

B.2.2.RoleMgr

Nome: RoleMgr

Intenção: Cuida do gerenciamento dos papéis atribuídos aos participantes.

Aplicabilidade: O gerenciamento de papéis possibilita a atribuição de permissões e tarefas específicas para determinadas funções dos participantes.

Variabilidade: Pode-se configurar o conjunto de papéis.

Estrutura:

O componente RoleMgr possui duas interfaces fornecidas (IRoleMgrMgr e IRoleMgrConfig) e uma interface requerida (Participant), conforme ilustrado na Figura B.8. A interface IRoleMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, atribuição dos papéis, recuperação dos participantes de um papel, etc. A interface IRoleMgrMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição do conjunto de papéis. O componente pode ser configurado através desta interface e por seu arquivo descritor.
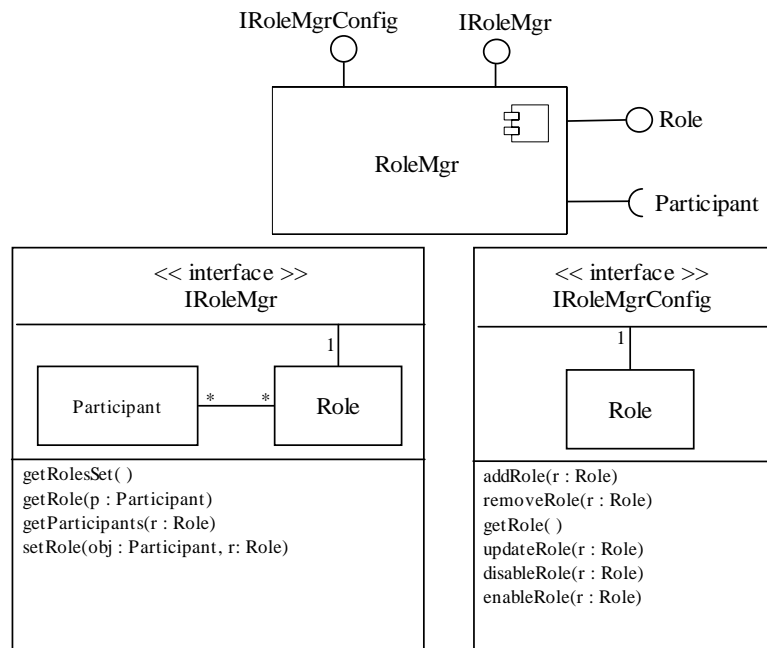
Figura B.8. Componente RoleMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet (funcionalidades específicas para os mediadores e aprendizes).

Componentes relacionados: ParticipantMgr

B.2.3.PermissionMgr

Nome: PermissionMgr

Intenção: Gerencia as permissões de execução de ações.

Aplicabilidade: Prover a segurança de acesso aos recursos e funcionalidades do sistema. Cada componente do sistema pode registrar neste componente as ações sobre as quais se deseja controlar permissões e as permissões de um determinado papel.

Variabilidade: Conjunto de ações de um serviço.

Estrutura:

O componente PermissionMgr possui duas interfaces fornecidas (IPermissionMgr e IPermissionMgrConfig), duas interfaces requeridas (Action e Role), conforme ilustrado na Figura B.9. A interface IPermissionMgr disponibiliza os serviços relativos à utilização do componente, como atribuição das permissões de um papel, referente às ações do serviço em questão, teste de

permissão, etc. A interface IPermissionMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição das ações. O componente pode ser configurado através desta interface e por seu arquivo descritor.
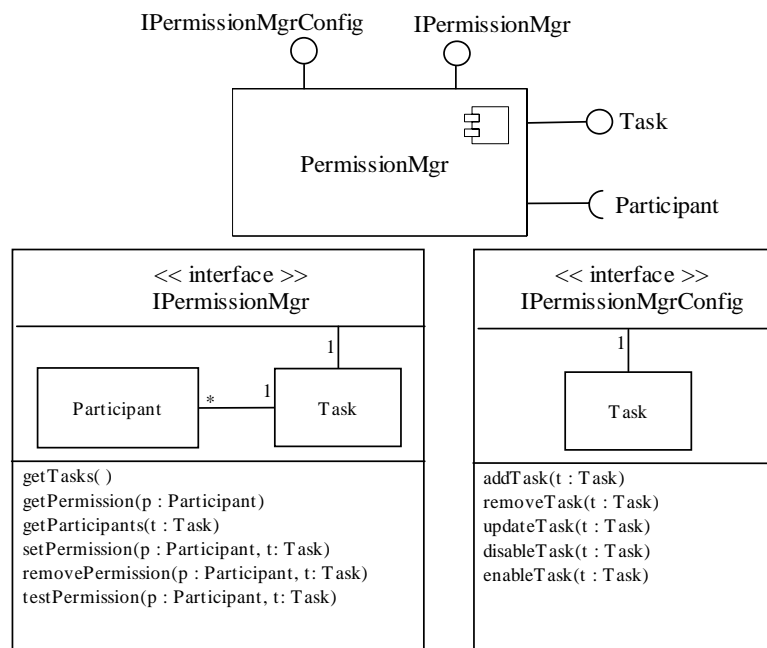
Figura B.9. Componente PermissionMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: -

B.2.4.ParticipantMgr

Nome: ParticipantMgr

Intenção: Gerencia os participantes da colaboração.

Aplicabilidade: O componente é utilizado para gerenciar a representação dos participantes.

Variabilidade: Pode-se configurar as informações que deseja-se registrar sobre os participantes.

Estrutura:

O componente ParticipantMgr possui três interfaces fornecidas (IParticipantMgr, IParticipantMgrConfig e Participant), conforme ilustrado na

Figura B.10. A interface IParticipantMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, criação, remoção e atualização de participante, busca, etc. A interface IParticipantMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição das informações a serem registradas para os participantes.
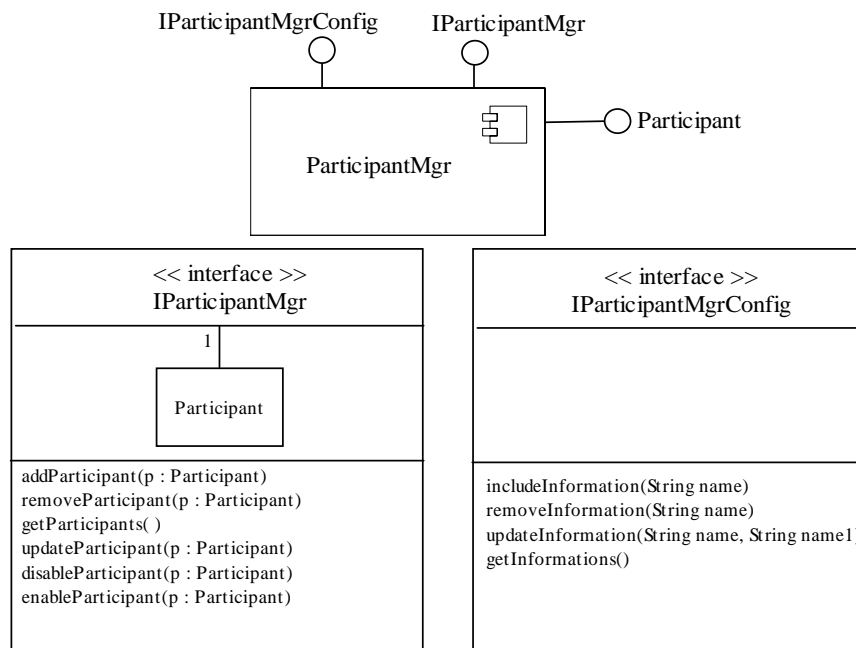
Figura B.10. Componente ParticipantMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: -

B.2.5.GroupMgr

Nome: GroupMgr

Intenção: Possibilita a definição e gestão de grupos e subgrupos de participantes.

Aplicabilidade: A colaboração pressupõe um grupo de participantes. Este componente possibilita organizar os participantes em grupos e subgrupos. Pode-se representar informações adicionais para o participante no escopo do grupo.

Variabilidade: tamanho do grupo

Estrutura:

O componente GroupMgr possui três interfaces fornecidas (IGroupMgr, IGroupMgrConfig e Group) e uma interface requerida (Participant), conforme ilustrado na Figura B.11. A interface IGroupMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, inclusão de participante em grupo, divisão de grupo, etc. A interface IGroupMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição dos grupos disponíveis. O componente pode ser configurado através desta interface e por seu arquivo descritor.
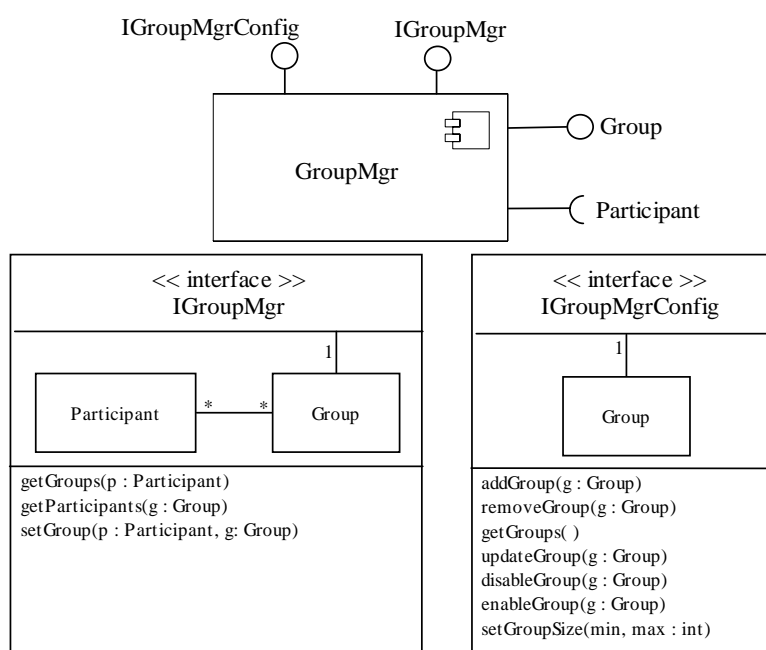
Figura B.11. Componente GroupMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: -

B.2.6.SessionMgr

Nome: SessionMgr

Intenção: Gerencia as sessões de colaboração.

Aplicabilidade: Possibilita definir a sessão, que agrega participantes, objetos compartilhados durante um período do tempo. É possível configurar a data de início e fim da sessão e o modo de entrada.

Variabilidade: Duração

Estrutura:

O componente SessionMgr possui três interfaces fornecidas (ISessionMgr, ISessionMgrConfig e Session) e requer as interfaces Group, Participant e CooperationObj, conforme ilustrado na Figura B.12. A interface ISessionMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, inclusão de participante ou objeto na sessão, verificação de disponibilidade, bloquear e desbloquear, etc. A interface ISessionMgrConfig disponibiliza os serviços relativos à configuração do componente, como criação da sessão e definição de suas propriedades. O componente pode ser configurado através desta interface e por seu arquivo descritor.
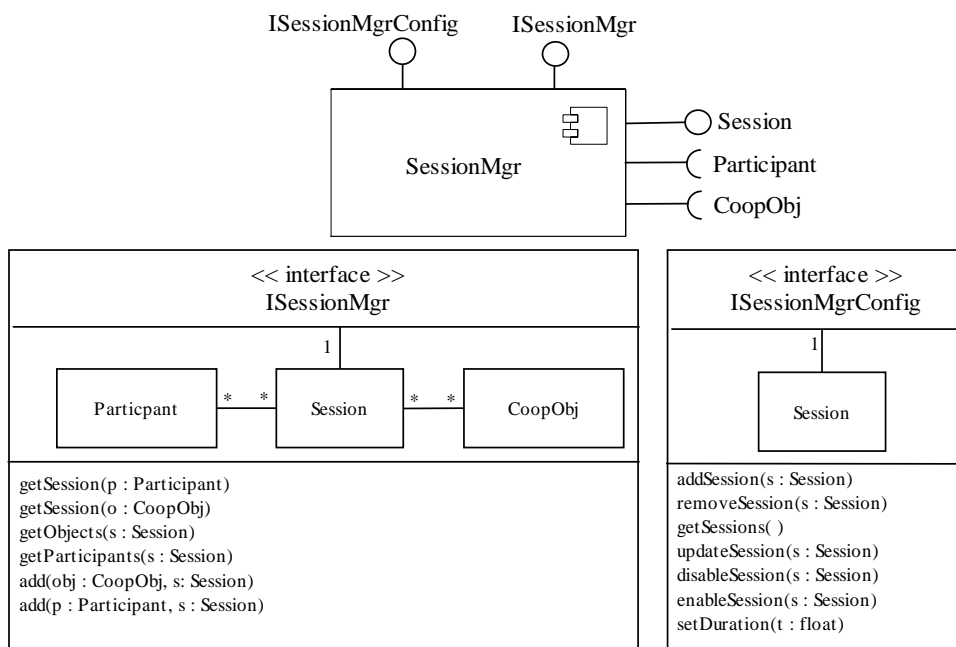
Figura B.12. Componente SessionMgr e suas interfaces

Usos conhecidos: Conferências, Debate, Correio para Turma e Correio para Participante.

Componentes relacionados: CooperationObjMgr, ParticipantMgr, GroupMgr.

B.2.7. FloorControlMgr

Nome: FloorControlMgr

Intenção: Gerencia a ordem de participação. Podem ser alocadas várias políticas de acesso.

Aplicabilidade: A definição da ordem de participação é útil para organizar a discussão ou a atuação em um espaço compartilhado.

Variabilidade: Pode-se a técnica desejada.

Estrutura:

O componente FloorControlMgr possui duas interfaces fornecidas (IFloorControlMgr e IFloorControlMgrConfig), conforme ilustrado na Figura B.13. A interface IFloorControlMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, recuperar a fila de participação, alterar a fila, alterar a técnica, etc. A interface IFloorControlMgrConfig disponibiliza os serviços relativos à configuração do componente, que incluem a definição das técnicas de participação. O componente pode ser configurado através desta interface e por seu arquivo descritor.
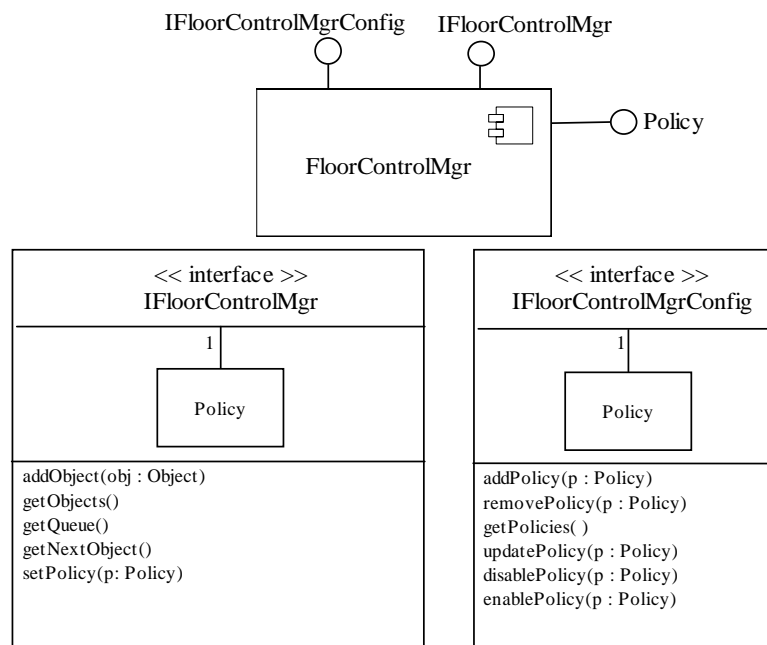
Figura B.13. Componente FloorControlMgr e suas interfaces

Usos conhecidos: Debate.

Componentes relacionados: -

B.2.8.TaskMgr

Nome: TaskMgr

Intenção: Possibilita o gerenciamento das tarefas do grupo.

Aplicabilidade: O gerenciamento das tarefas possibilita o acompanhamento da realização e da produtividade dos participantes.

Variabilidade: Podem ser configurados os tipos de tarefas.

Estrutura:

O componente TaskMgr possui duas interfaces fornecidas (ITaskMgr e ITaskMgrConfig) e duas interfaces requeridas (Task e Participant), conforme ilustrado na Figura B.14. A interface ITaskMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, atribuição de tarefas a participantes, mudança de status de tarefa, etc. A interface ITaskMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição do conjunto de tarefas e seus tipos. O componente pode ser configurado através desta interface e por seu arquivo descritor.
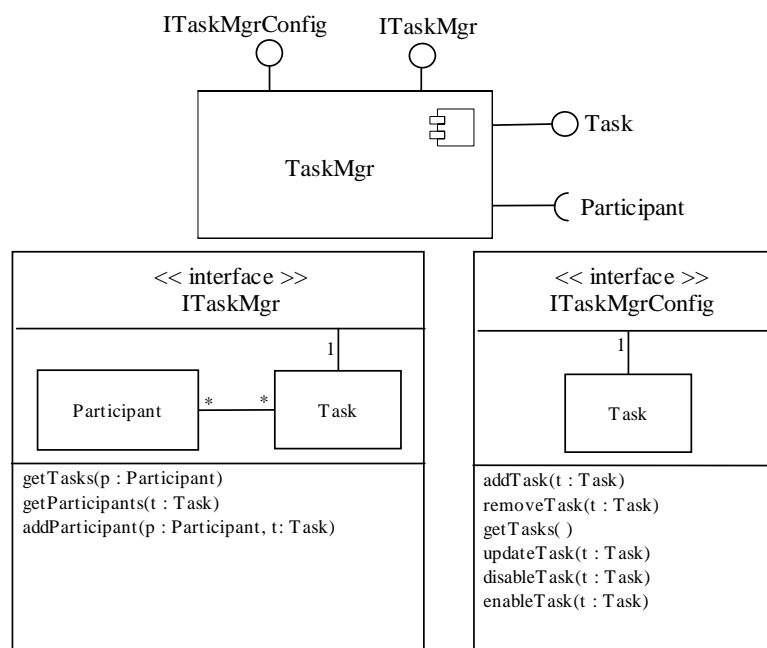
Figura B.14. Componente TaskMgr e suas interfaces

Usos conhecidos: Serviço Tarefas.

Componentes relacionados: ParticipantMgr

B.2.9.AwarenessMgr

Nome: AwarenessMgr

Intenção: Gerencia as informações de percepção em geral, registrando e operando sobre os eventos ocorridos no serviço.

Aplicabilidade: As informações de percepção contextualizam o trabalho em grupo. Alguns exemplos de informações: o que já foi feito, o que falta fazer, novidades, presença de participante, novidades desde a última visita, etc. As informações de percepção são especialmente úteis para embasar filtros utilizados na computação móvel, visto que a informação deve ser mais precisa e focada, dadas as restrições de tamanho de tela, banda e qualidade de conexão [Filippo et al., 2005].

Variabilidade: Pode-se configurar os tipos de eventos.

Estrutura:

O componente AwarenessMgr possui quatro interfaces fornecidas (IAwarenessMgr, IAwarenessMgrConfig, Event e EventType) e uma interface requerida (Participant), conforme ilustrado na Figura B.15. A interface IAwarenessMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, criação e recuperação de eventos. A interface IAwarenessMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição dos tipos de eventos. O componente pode ser configurado através desta interface e por seu arquivo descritor.
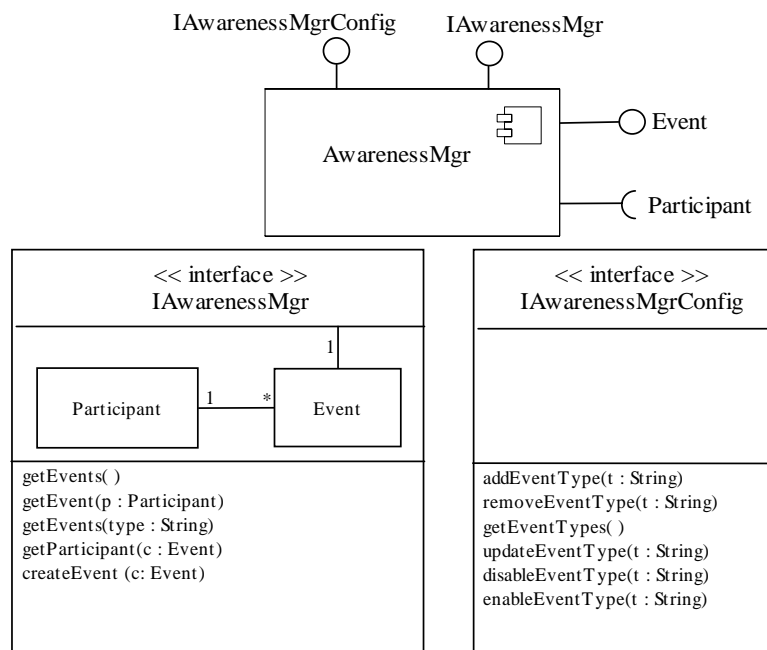
Figura B.15. Componente AwarenessMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: ParticipantMgr

B.2.10.AvailabilityMgr

Nome: AvailabilityMgr

Intenção: Possibilita que o participante configure sua disponibilidade.

Aplicabilidade: Saber a disponibilidade dos participantes contribui para sua melhor organização, diminuindo a quantidade de vezes que os participantes são interrompidos em horas impróprias.

Variabilidade: Pode-se configurar os graus de disponibilidade.

Estrutura:

O componente AvailabilityMgr possui duas interfaces fornecidas (IAvailabilityMgr e IAvailabilityMgrConfig) e uma interface requerida (Participant), conforme ilustrado na Figura B.16. A interface IAvailabilityMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, mudança de disponibilidade, consulta de disponibilidade, etc. A interface IAvailabilityMgrConfig disponibiliza os serviços relativos à configuração do

componente, incluindo a definição dos níveis de disponibilidade. O componente pode ser configurado através desta interface e por seu arquivo descritor.
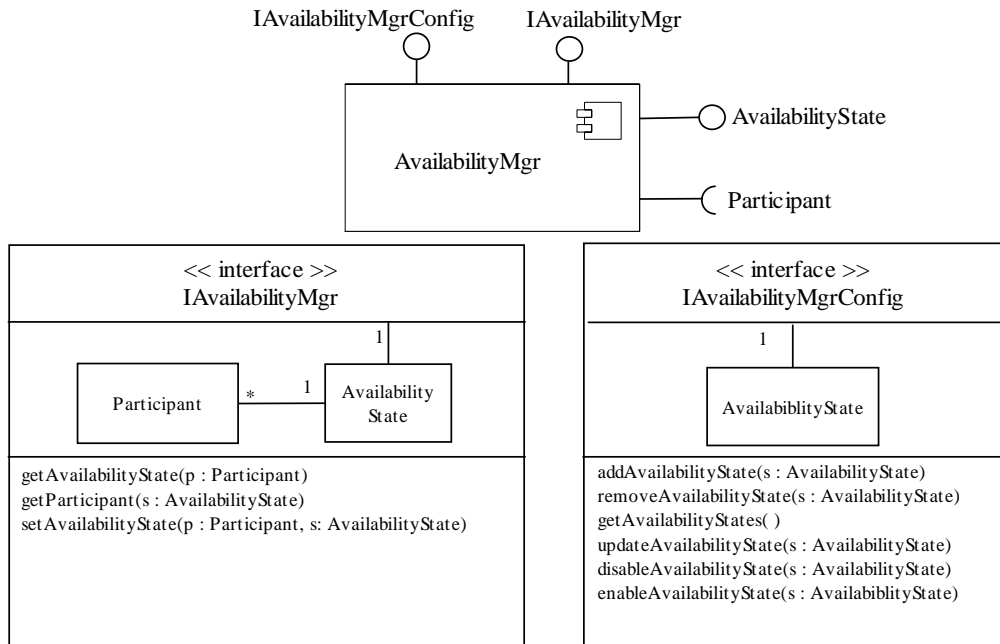
Figura B.16. Componente AvailabilityMgr e suas interfaces

Usos conhecidos: -

Componentes relacionados: ParticipantMgr

B.2.11.NotificationMgr

Nome: NotificationMgr

Intenção: Gerencia o envio das notificações aos participantes.

Aplicabilidade: Um canal único de notificação para os diversos serviços favorece o estabelecimento de modos de recebimento para cada participante e filtros específicos.

Variabilidade: Pode-se configurar os tipos de notificações.

Estrutura:

O componente NotificationMgr possui duas interfaces fornecidas (INotificationMgr e INotificationMgrConfig) e uma interface requerida (Participant), conforme ilustrado na Figura B.17. A interface INotificationMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo,

envio de notificação, recuperação de notificações anteriores, etc. A interface INotificationMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição dos tipos de notificações, cabeçalho genérico, etc. O componente pode ser configurado através desta interface e por seu arquivo descritor.
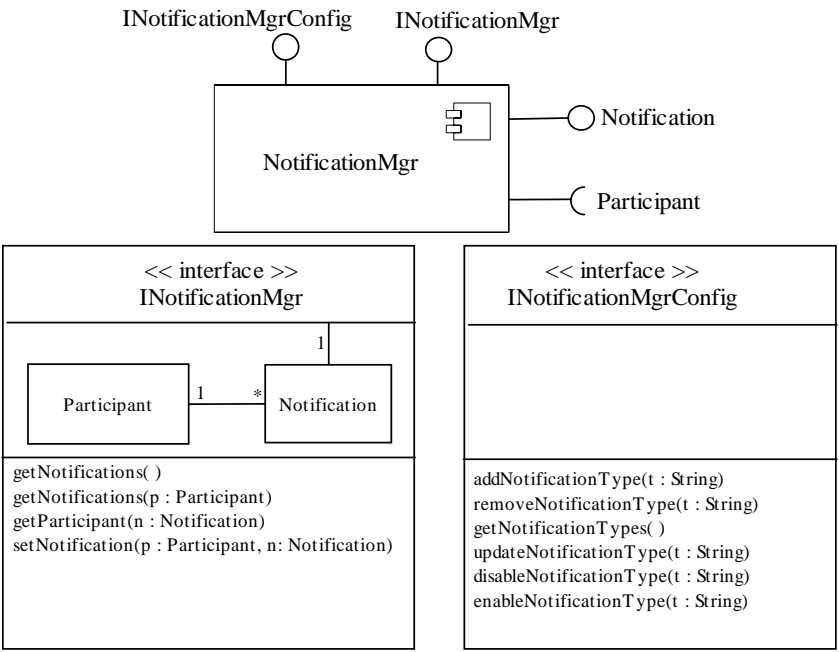
Figura B.17. Componente NotificationMgr e suas interfaces

Usos conhecidos: Conferências e Correio para Turma.

Componentes relacionados: ParticipantMgr

B.3.Componentes de Cooperação

Os componentes de cooperação oferecem suporte aos objetos compartilhados e sua manipulação. A seguir, são descritos os componentes CooperationObjMgr, SearchMgr, StatisticalAnalysisMgr, ActionLogMgr, AccessRegistrationMgr.

B.3.1.CooperationObjMgr

Nome: CooperationObjMgr

Intenção: Provê mecanismos de compartilhamento e concorrência aos objetos compartilhados. Possibilita também a persistência dos objetos.

Aplicabilidade: Os objetos compartilhados são manipulados ao longo da colaboração. Este componente oferece suporte a gestão destes objetos.

Variabilidade: -

Estrutura:

O componente CooperationObjMgr possui três interfaces fornecidas (ICooperationObjMgr e ICooperationObjMgrConfig) e uma interface requerida (Participant), conforme ilustrado na Figura B.18. A interface ICooperationObjMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, criação e recuperação dos objetos. A interface ICooperationObjMgrConfig disponibiliza os serviços relativos à configuração do componente. O componente pode ser configurado através desta interface e por seu arquivo descritor.
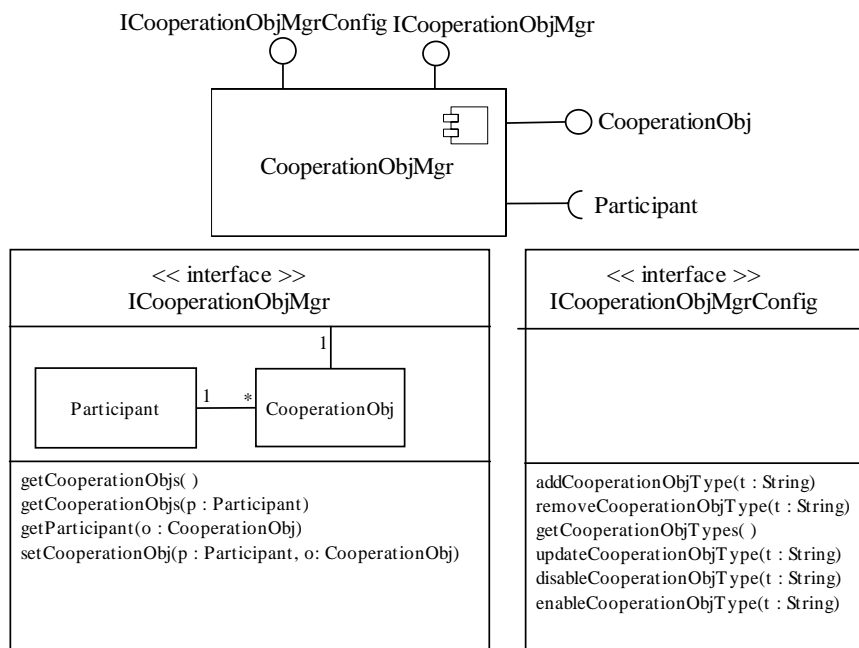
Figura B.18. Componente CooperationObjMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: ParticipantMgr

B.3.2. SearchMgr

Nome: SearchMgr

Intenção: Provê mecanismos de busca para os objetos compartilhados.

Aplicabilidade: Possibilitar utilizar critérios de busca para recuperar os objetos de cooperação de um repositório.

Variabilidade: Critérios de busca

Estrutura:

O componente SearchMgr possui duas interfaces fornecidas (ISearchMgr e ISearchMgrConfig) e uma interface requerida (CooperationObj), conforme ilustrado na Figura B.19. A interface ISearchMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, recuperação dos objetos. A interface ISearchMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição do conjunto de critérios disponívies. O componente pode ser configurado através desta interface e por seu arquivo descritor.
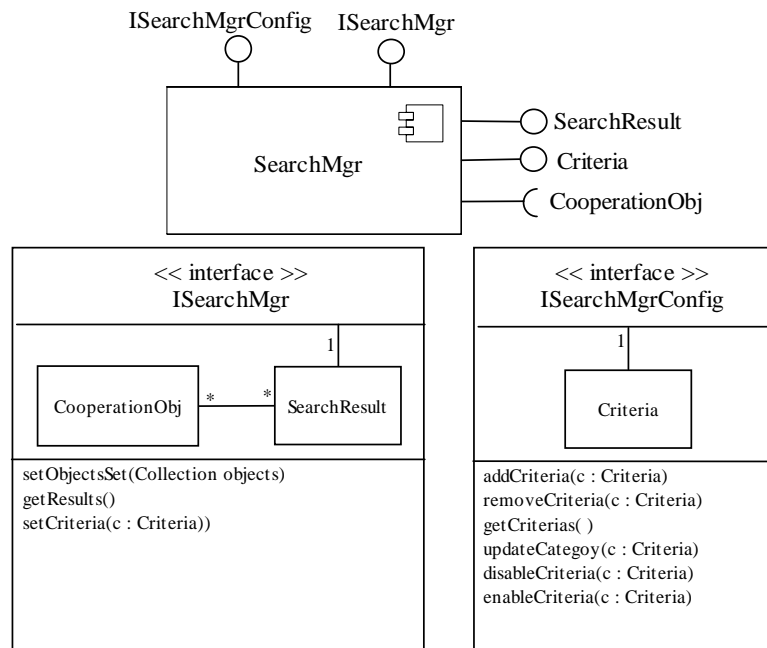
Figura B.19. Componente SearchMgr e suas interfaces

Usos conhecidos: Conferências.

Componentes relacionados: CooperationObjMgr

B.3.3.StatisticalAnalysisMgr

Nome: StatisticalAnalysisMgr

Intenção: Oferece funcionalidades de análise estatística dos objetos compartilhados.

Aplicabilidade: Oferece análises estatísticas sobre os objetos registrados, de modo a embasar a toma de decisão e a coordenação do grupo [Gerosa et al., 2005].

Variabilidade: -

Estrutura:

O componente StatisticalAnalysisMgr possui duas interfaces fornecidas (IStatisticalAnalysisMgr e IStatisticalAnalysisMgrConfig), conforme ilustrado na Figura B.20. A interface IStatisticalAnalysisMgr disponibiliza os serviços relativos à utilização do componente e a interface IStatisticalAnalysisMgrConfig disponibiliza os serviços relativos à configuração do componente. O componente pode ser configurado através desta interface e por seu arquivo descritor.
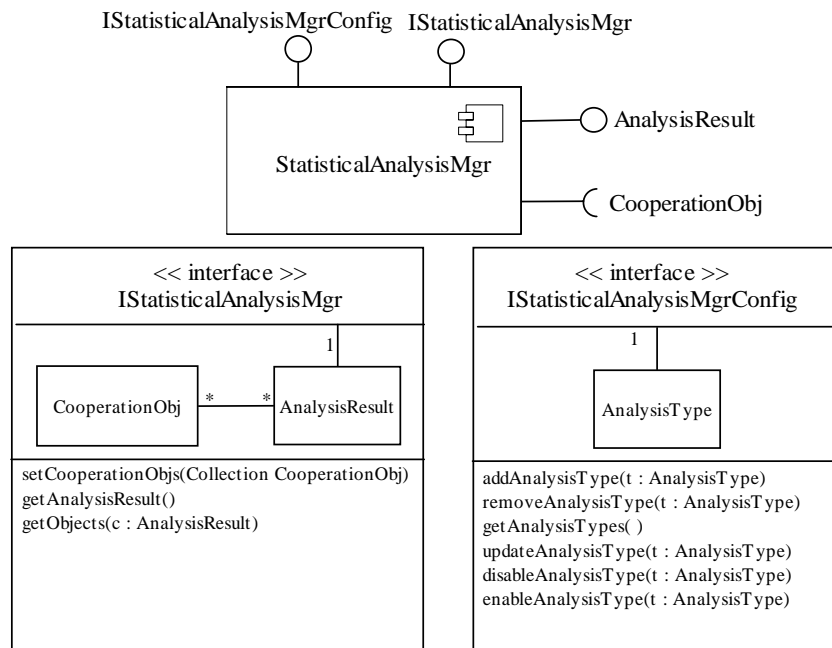
Figura B.20. Componente StatisticalAnalysisMgr e suas interfaces

Usos conhecidos: Conferências e Correio para Turma.

Componentes relacionados: CooperationObjMgr

B.3.4.ActionLogMgr

Nome: ActionLogMgr

Intenção: Possibilita registrar o histórico de ações no serviço.

Aplicabilidade: O registro do log de ações possibilita fazer auditoria, acompanhar o uso, voltar a situações anteriores, rastrear problemas, otimizar processos, etc.

Variabilidade: Pode ser configurado o nível e o tipo de log a ser gerado.

Estrutura:

O componente ActionLogMgr possui duas interfaces fornecidas (IActionLogMgr e IActionLogMgrConfig), conforme ilustrado na Figura B.21. A interface IActionLogMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, registro e recuperação de ações. A interface IActionLogMgrConfig disponibiliza os serviços relativos à configuração do componente, como definição dos tipos e níveis de log. O componente pode ser configurado através desta interface e por seu arquivo descritor.
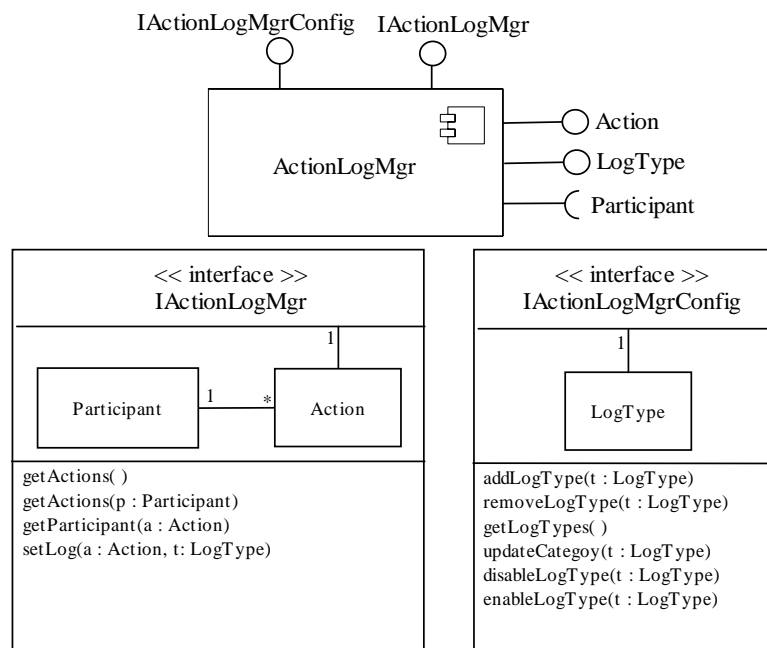
Figura B.21. Componente ActionLogMgr e suas interfaces

Usos conhecidos: Todos os serviços do AulaNet.

Componentes relacionados: ParticipantMgr

B.3.5.AccessRegistrationMgr

Nome: AccessRegistrationMgr

Intenção: Registra os acessos dos participantes para cada objeto, possibilitando um controle do que já foi visitado.

Aplicabilidade: Registrando o acesso aos objetos é possível exibir diferentemente os conteúdos não visitados, embasando a navegação dos participantes.

Variabilidade: -

Estrutura:

O componente AccessRegistrationMgr possui duas interfaces fornecidas (IAccessRegistrationMgr e IAccessRegistrationMgrConfig) e duas interfaces requeridas (Participant e CooperationObj), conforme ilustrado na Figura B.22. A interface IAccessRegistrationMgr disponibiliza os serviços relativos à utilização do componente, como por exemplo, registro e recuperação dos acessos. A interface IAccessRegistrationMgrConfig disponibiliza os serviços relativos à

configuração do componente. O componente pode ser configurado através desta interface e por seu arquivo descritor.
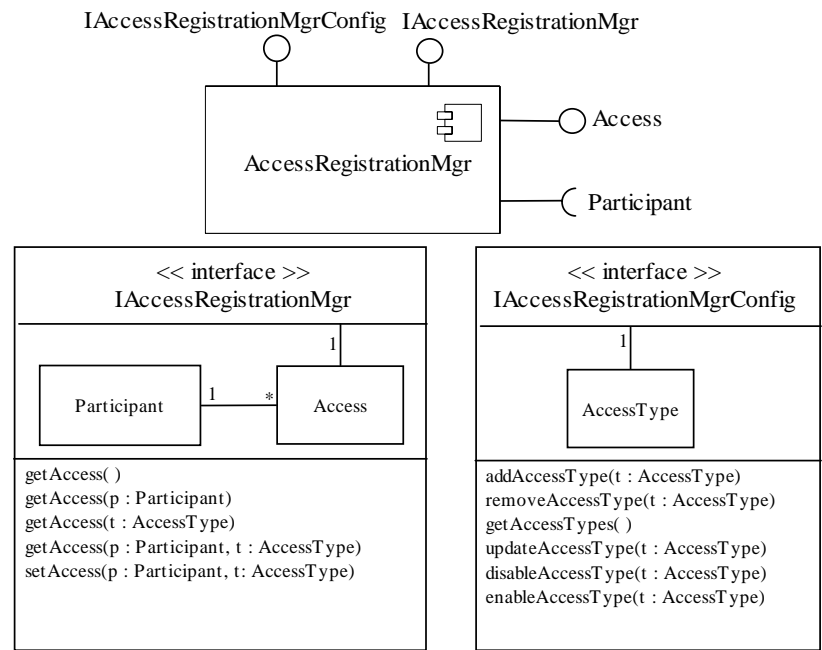
Figura B.22. Componente AccessRegistrationMgr e suas interfaces

Usos conhecidos: Conferências, Correio para Turma e Correio para Participante

Componentes relacionados: ParticipantMgr

Referências Bibliográficas

Allen, J. F. (1984) Towards a General Theory of Action and Time. Artificial Intelligence, 23, 1984, 123-154.

Alur D., Crupi, J. & Malks, D. (2001) Core J2EE Patterns: Best Practices and Design Strategies. Publisher: Prentice Hall / Sun Microsystems Press, 2001.

Amiour, M. & Estublier, J. (1998) A Support for Communication in Software Processes. 10th Conference on Software Engineering and Knowledge Engineering (SEKE'98).

Amiour, M. (1997) "A Support for cooperation in software processes". Doctoral Consortium of the 9th Conference on Advanced Information Systems Engineering (CAiSE'97), Barcelona, Spain.

Anderson, G.E., Graham, T.C. & Wright, T.N. (2000) "Dragonfly: linking conceptual and implementation architectures of multiuser interactive systems", Proceedings of the 22nd international Conference on Software Engineering (ICSE '00), Limerick, Ireland, June 04 - 11, 2000, ACM Press, pp. 252-261.

Andrade, L.V., Sampaio, F.F. & Rocha, L.L.A. (2002) O Modelo GD-IBIS: Grupo de Discussão para Web no Contexto de Educação a Distâncial, XXIX SEMISH, Anais do XXII Congresso da Sociedade Brasileira de Computação. Volume 3, pp. 183-292.

Apperly, H. (2001) The Component Industry Metaphor, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Arango, G. (1994) Domain Analysis Methods. In: Software Reusability, Schäfer, W., Prieto-Diaz, R. & Matsumoto, M. (eds), Chichester, Ellis Horwood, 1994, pp. 17-49.

Araujo, R.M., Santoro, F.M. & Borges, M.R.S. (2004) "A conceptual framework for designing and conducting groupware evaluations", Journal of Computer Applications in Technology (IJCAT), V. 19, N. 3-4, Special Issue on Current Approaches for Groupware Design, Implementation and Evaluation, pp. 139-150.

Araujo, R.M., Santoro, F.M. & Borges, M.R.S. (2004) "The CSCW Lab Ontology for Groupware Evaluation", Proceedings of the 8th Computer Supported Cooperative Work in Design (CSCWiD 2004), pp. 148-153.

Baker, K., Greenberg, S. & Gutwin, C. (2001) Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration. 8th IFIP International Conference (EHCI 2001). Lecture Notes in Computer Science Vol. 2254, pp. 123-139, Springer-Verlag.

Banavar, G., Doddapaneti, S., Miller, K. & Mukherjee, B. (1998) Rapidly Building Synchronous Collaborative Applications by Direct Manipulation. In Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCW'98), pp. 139-148, 1998.

Bandinelli, S., Nitto, E.D. & Fuggetta, A. (1996) “Supporting cooperation in the SPADE-1 Environment”, IEEE Transactions on Software Engineering, vol. 22, No. 12, December 1996, pp. 841-865

Barreto, C.G. (2006) Agregando Frameworks de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet, Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, 2006.

Barreto, C.G., Fuks, H. & Lucena, C.J.P. (2005) “Agregando Frameworks em uma Arquitetura Baseada em Componentes no Ambiente AulaNet”, Anais do 5º Workshop de Desenvolvimento Baseado em Componentes - WDBC 2005, 7-9 de novembro de 2005, Juiz de Fora, MG, ISBN 85-88279-47-9, pp. 25-32.

Barroca, L., Gimenes, I.M.S. & Huzita, E.H.M. (2005) “Conceitos Básicos”, in: Desenvolvimento Baseado em Componentes, Gimenes, I.M.S. & Huzita, E.H.M. (eds), Editora Ciência Moderna, Rio de Janeiro, 2005. ISBN 85-7393-406-9, pg. 57-103.

Barros, L.A. (1994) Suporte a Ambientes Distribuídos para Aprendizagem Cooperativa, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

Bass, L., Clements, P. & Kazman, R. (2003) Software Architecture in Practice, Addison-Wesley 2003. ISBN: 0-321-15495-9

Beck, K. (2004): Programação extrema explicada: acolha as mudanças. Porto Alegre: Bookman.

Becker, K. & Zanella, A.N. (1998) “A Cooperation Model for Teaching/Learning Modeling Disciplines”, International Workshop on Groupware (CRIWG 1998), Brasil.

Begole, J.B., Rosson, M.B. & Shaffer, C.A. (1999). Flexible collaboration transparency: Supporting worker independence in replicated application-sharing systems. ACM Transactions on Computer–Human Interaction, 6(2), 95–132.

Benbunan-Fich, R. & Hiltz, S. R. (1999) Impacts of Asynchronous Learning Networks on Individual and Group Problem Solving: A Field Experiment, Group Decision and Negotiation, Vol.8, pp. 409-426.

Blikstein, I. (2000) Técnicas de comunicação escrita. Coleção Princípios, Ed. Ática, ISBN 8508094884.

Blois, A.P.T.B. & Becker, K.A. (2002) “Component-based Architecture to Support Collaborative Application Design”, 8th International Workshop on Groupware (CRIWG), LNCS Vol. 2440, Springer-Verlag, p. 134-146

Blois, A.P.T.B., Werner, C.M.L. & Becker, K. (2004) “Um Processo de Engenharia de Domínio com foco no Projeto Arquitetural Baseado em Componentes”, IV Workshop de Desenvolvimento Baseado em Componentes, João Pessoa, PB, pp. 15-20.

Blois, M., Choren, R., Laufer, C., Ferraz, F. & Fuks, H. (1999) “Desenvolvendo Aplicativos para a Web com o Scriba”, Anais do XXVI SEMISH - Seminário Integrado de Software e Hardware, Sociedade Brasileira de Computação (SBC), Rio de Janeiro, pp 119-133.

Bloom, B.S. (1956) “Taxonomy of educational objectives: handbook 1, cognitive domain”, New York, Longman.

Boehm, B.W. (1988) A Spiral Model of Software Development and Enhancement, IEEE Computer, Vol. 21, No. 5, pp. 61-72

Booch, G. (1987) Software Components with Ada: Structures, Tools, and Subsystems. Benjamin-Cummings, Redwood City, CA.

Booch, G., Rumbaugh, J. & Jacobson, I. (2000) UML: Guia do Usuário. Editoria Campus, Rio de Janeiro. ISBN 85-352-0562-4

Borges, M.R. & Pino, J.A. (1999) “Awareness mechanisms for coordination in asynchronous CSCW,” Procs. of the 9th Workshop on Information Technologies and Systems (WITS'99), 1999, pp. 69-74.

Borges, M.R.S., Brézillon, P., Pino, J.A. & Pornerol, J.C. (2004) “Bringing Context to CSCW”, Proceedings of The 8th International Conference on Computer Supported Cooperative Work in Design (CSCWiD 2004), China, IEEE, pp. 161-166.

Borges, M.R.S., Mendes, S. & Motta, C.L.R. (2002) “Improving Meetings by identifying informal roles played by participants”, 7th International Conference on Computer Supported Cooperative Work in Design, 2002, Rio de Janeiro, pp. 368-372.

Borges, M.R.S., Pino, J.A. & Salgado, A.C. (2000) Requirements for Shared Memory in CSCW Applications, Proceedings of the 10th Workshop on Information Technology and Systems (WITS’00), Brisbane, Australia.

Borghoff, U.M. & Schlichter, J.H. (2000) Computer-Supported Cooperative Work: Introduction to Distributed Applications. Springer, USA.

Braga, R. (2000) “Busca e Recuperação de Componentes em Ambientes de Reutilização de Software”. Tese de Doutorado. COPPE Sistemas, 2000.

Bretain, I., Fredin, L., Frost, W., Hedman, L.R., Kroon, P., McGlashan, S., Sallnas, E.L. & Virtanen, M. (1997) Leave the Office, Bring Your Colleagues: Design Solutions for Mobile Teamworkers. Proc. CHI’97, ACM Press, pp.335-336

Brna, P. (1998) “Modelos de colaboração”, Revista Brasileira de Informática e Educação, 3, pp. 1-15.

Brockschmidt, K. (1996) What OLE Is Really About, MSDN Archive, Microsoft Comporation. msdn.microsoft.com/archive/en-us/dnarolegen/html/msdn_aboutole.asp

Brown, A.W. (1996) Component-Based Software Engineering. IEEE Computer Society.

Buzko, D., Lee, W. & Helal, A. (2000) Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration. In Proceedings of Group 2001, Collaborative Editing Workshop.

Calvary, G., Coutaz, J. & Nigay, L. (1997) From Single-User Architectural Design to PAC*: a Generic Software Architectural Model for CSCW. Conference on Human Factors in Computing Systems (CHI’97), pp 242-249.

Castellani, S., Ciancarini, P. & Rossi, D. (1996) “The ShaPE of ShaDE: a Coordination System”, Technical Report UBLCS 96-5, Dipartimento di Scienze dell’Informazione, Università di Bologna, Italy, March, 1996.

Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S. & Seguin, C. (1998) Java Object-Sharing in Habanero. Communications of the ACM, Vol. 41 # 6, June 1998.

Cheesman, J. & Daniels, J. (2000) UML Components: A SimpleProcess for Specifying Component-Based Software, Addison-Wesley, Reading, Massachusetts.

Chung, G. & Dewan, P. (2004) "Towards Dynamic Collaboration Architectures", Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'04), November 6-10, Volume 6, Issue 3.

Churcher, N. & Cerecke, C. (1996) "GroupCRC: Exploring CSCW support for software engineering", Proceedings of the 6th Australian conference on computer-human interaction, Hamilton, New Zealand, November 24-27, 1996. IEEE Computer Society Press, Los Alamitos, CA, USA, 1996, pp. 62-68.

Clements, P. & Northrop, L. (2001) Software Product Lines – Practices and Patterns. Readding, Addison-Wesley, 2001.

Conklin, J. & Begeman, M. (1988) "gIBIS: A hypertext tool for exploratory policy discussion", ACM Transactions on Office Information Systems, Vol. 3, No. 3

Councill, B. & Heineman, G.T. (2001) Definition of a Software Component and Its Elements, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Cousineau, G. & Mauny, M. (1998) The Functional Approach to Programming, Cambridge University Press, English edition, December 1998. ISBN: 0521576814

Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2003) "A Adaptação do Ambiente AulaNet para Dar Suporte a Grupos de Aprendizagem e sua Formação Utilizando os Conceitos de Agentes de Software", Revista Brasileira de Informática na Educação, ISSN 1414-5685, Sociedade Brasileira de Computação, V. 11, No. 1, Abril de 2003, pp. 26-46.

D'Souza, D.F. & Wills, A.C. (1998) Objects, Components and Frameworks with UML: The Catalysis Approach. Addison Wesley, ISBN 0-201-31012-0, 1998.

Daft, R.L. & Lengel, R.H. (1986). Organizational information requirements, media richness and structural design. Management Science 32(5), 554-571.

David, J.M.N. & Borges, M.R.S. (2001) "Selectivity of awareness components in asynchronous CSCW environments", Proceedings of 7th International Workshop on Groupware (CRIWG 2001), IEEE, Darmstadt, Germany.

David, J.M.N. & Borges, M.R.S. (2004) "Designing collaboration through a web-based groupware infrastructure", International Journal of Computer Applications in Technology, v.19, n.3/4, pp. 175-183.

Dewan, P. (1998) Architectures for Collaborative Applications. In M.Beaudouin-Lafon (Ed.), Computer Supported Cooperative Work (CSCW) (7 ed., pp. 169-194). John Wiley & Sons Ltd.

Dias, M.S. & Borges, M.R.S. (1999) "Development of groupware systems with the COPSE infrastructure", International Workshop on Groupware (CRIWG 1999), Cancun, IEEE Computer Society, pp. 278-285.

Dillenbourg, P. & Self,. J.A. (1992) "A computational approach to socially distributed cognition", European Journal of Psychology of Education, Vol VII, No 4, pp 252-273, 1992.

Dillenbourg, P. (1999) “What do you mean by collaborative learning?”, Collaborative-learning: Cognitive and Computational Approaches, Oxford, Elsevier, 1999, pp. 1-19

Dourish, P. & Belloti, V. (1992) “Awareness and coordination in shared workspaces,” Proceedings of CSCW’92, Chapel Hill NC, 1992.

Dourish, P. (1998) Using Metalevel techniques in a flexible toolkit for CSCW applications, ACM Transactions on Computer-Human Interaction, Vol 5, No 2, pp. 109-155.

Dron J., Boyne C. & Mitchell R. (2001) “Footpaths in the stuff swamp” Proceedings of WebNet'2001, Fowler, W. & Hasebrook, J. (eds.), WorldConference of the WWW and Internet, Orlando, FL, AACE, pp. 323-328.

Durfee, E.H. (1988) Coordination of Distributed Problem Solvers, The International Series in Engineering and Computer Science, Vol. 55, ISBN: 0-89838-284-X.

Edutools (2005) http://www.edutools.info (date 06/04/2005)

Ellis, C.A. & Wainer, J. (1994) A Conceptual Model of Groupware, In T. Malone (ed) Conference on Computer-Supported Cooperative Work (CSCW), pp. 79-88.

Ellis, C.A., Gibbs, S.J. & Rein, G.L. (1991) Groupware - Some Issues and Experiences. Communications of the ACM, Vol. 34, No. 1, pp. 38-58.

Engelbart, D. & English, W. (1968) Research Center for Augmenting Human Intellect, Proc. Fall Joint Computing Conference, AFIPS Press, 395-410

Farias, C.R.G. (2004) Um Metamodelo para Sistemas Cooperativos. In: Workshop Brasileiro de Tecnologias para Colaboração, Ribeirão Preto. Proceedings of WebMedia & LA-Web 2004 Joint Conference, 2004. v. 2. pp. 194-201.

Fayad, M. E. & Johnson, R. E. (2000): Domain-specific application frameworks: frameworks experience by industry. New York: J. Wiley, c2000. 681 p. ISBN 0471332801.

Fayad, M. E. & Schmidt, D. C. (1997): Object-oriented application frameworks, Communications of the ACM, v.40 n.10, p.32-38, Oct. 1997

Fayad, M. E., Schimidt & D. C., Johnson, R. E. (1999a): Implementing application frameworks: object-oriented frameworks at work. New York: J. Wiley, c1999. 729 p. ISBN 0471252018.

Fayad, M. E., Schimidt & D. C., Johnson, R. E. (1999b): Building application frameworks: object-oriented foundations of framework design. New York: J. Wiley, c1999. 664 p. ISBN 0471248754.

Fielding, R.T. (1999) Shared leadership in the Apache project, Communications of the ACM, Volume 42 , Issue 4, pp 42-43.

Filippo, D., Fuks, H. & Lucena, C.J.P. (2005) AulaNetM: Extensão do Serviço de Conferências do AulaNet destinada a usuários de PDAs. Anais do XVI Simpósio Brasileiro de Informática na Educação - SBIE 2005, Juiz de Fora, MG, 9 a 11 de novembro de 2005, pp. 623-633

Fitzpatrick, G., Kaplan, S., Mansfield, T. Arnold, D. & Segall, B. (2002) Supporting Public and Accessibility with Elvin: Experiences and Reflections. Computer Supported Cooperative Work, Vol. 11, No. 3-4, pp. 447-474.

Fitzpatrick, G., Tolone, W.J. & Kaplan, S. M. (1995) Work, Locales and Distributed Social Worlds. In Proceedings of the 4th European Conference on Computer Supported Cooperative Work (ECSCW '95), pp. 1-16, 1995.

Fowler, M. (1996) Analysis Patterns: Reusable Object Models, Addison-Wesley.

Fowler, M. (2002) Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

Fuggetta, A. (2000) Software Process: A Roadmap, 22nd International Conference on Software Engineering – ICSE 2000, Future of Software Engineering Track.

Fuks, H. (2000) “Aprendizagem e Trabalho Cooperativo no Ambiente AulaNet”, Revista Brasileira de Informática na Educação, N6, Abril 2000, ISSN 1414-5685, Sociedade Brasileira de Computação, pp 53-73, 2000.

Fuks, H., Cunha, L.M., Gerosa, M.A. & Lucena, C.J.P. (2003) “Participação e Avaliação no Ambiente Virtual AulaNet da PUC-Rio”. in: Silva, M.; Educação Online: Teorias, Práticas, Legislação e Formação Corporativa; Edições Loyola, Rio de Janeiro, 2003, ISBN 85-15-02822-0, Cap. 15, pp. 231-254.

Fuks, H., Gerosa, M.A. & Lucena, C.J.P. (2002) “The Development and Application of Distance Learning on the Internet”. Open Learning - The Journal of Open and Distance Learning, Vol. 17, N. 1, ISSN 0268-0513, pp 23-38.

Fuks, H., Gerosa, M.A. & Pimentel, M. (2003) “Projeto de Comunicação em Groupware: Desenvolvimento, Interface e Utilização”. XXII Jornada de Atualização em Informática, Anais do XXIII Congresso da Sociedade Brasileira de Computação, V2, Cap. 7, ISBN 85-88442-59-0, pp. 295-338.

Fuks, H., Raposo, A.B. & Gerosa, M.A. (2002) “Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas”, XXI Jornada de Atualização em Informática, Anais do XXII Congresso da Sociedade Brasileira de Computação, V2, Cap. 3, ISBN 85-88442-24-8, pp. 89-128.

Fuks, H., Raposo, A.B., Gerosa, M.A. & Lucena, C.J.P. (2005) Applying the 3C Model to Groupware Development. International Journal of Cooperative Information Systems (IJCIS), v.14, n.2-3, Jun-Sep 2005, World Scientific, ISSN 0218-8430, pp. 299-328.

Funaro, G.M. & Montell, F. (1999) “Pedagogical roles and implementation guidelines for online communication tools”, ALN Magazine Volume 3, Issue 2, December 1999

Fussell, S.R., Kraut, R. E., Learch, F.J., Scherlis, W.L., McNally, M.M. & Cadiz, J.J. (1998) “Coordination, overload and team performance: effects of team communication strategies”, Proceedings of CSCW '98, Seattle, USA, p. 275-284. ISBN 1-58113-009-0

Gaines, B. (1999) “Modeling and forecasting the information sciences”, *Information Sciences* 57/58, pp. 13-22.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1994) Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, ISBN 0201633612.

Gandon, F. & Sadeh, N. (2004) Semantic Web Technology to Reconcile Privacy and Context Awareness, Web Semantics Journal. Vol. 1, No. 3, 2004.

Gerosa, M.A. (2002) “Categorização e Estruturação de Mensagens Textuais em Ambientes Virtuais de Colaboração”, Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 28 de fevereiro de 2002.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2001) “Um groupware baseado no ambiente AulaNet desenvolvido com componentes”, Anais eletrônicos do 1º Workshop de Desenvolvimento Baseado em Componentes, 21-22 Junho, Maringá-PR.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001) “Use of Categorization and Structuring of Messages in order to Organize the Discussion and Reduce Information Overload in Asynchronous Textual Communication Tools”. 7th International Workshop on Groupware (CRIWG 2001), M. Borges, J. Haake and U. Hoppe (eds.), IEEE, 6-8 September, Darmstadt - Germany, pp 136-141

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2002) “Resultados da avaliação de um curso baseado na Web”, VIII Workshop de Informática na Escola (WIE 2002), XXII Congresso da Sociedade Brasileira de Computação, V. 5, 17 a 19 de julho, Florianópolis, pp 477-485.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2003) Analysis and Design of Awareness Elements in Collaborative Digital Environments: A Case Study in the AulaNet Learning Environment. The Journal of Interactive Learning Research, 14(3), ISSN: 1093-023X, Association for the Advancement of Computing in Education, USA, pp. 315-332.

Gerosa, M.A., Pimentel, M., Fuks, H. & Lucena, C.J.P. (2005) “No Need to Read Messages Right Now: Helping Mediators to Steer Educational Forums Using Statistical and Visual Information”. Computer Supported Collaborative Learning, Taiwan, July 2005, ISBN 0805857826, Lawrence Erlbaum Associates, pp. 160-169.

Gimenes, I.M.S. & Huzita, E.H.M. (2005) Desenvolvimento Baseado em Componentes, Editora Ciência Moderna, Rio de Janeiro, 2005. ISBN 85-7393-406-9.

Goldratt, E.M. (1997) “Critical Chain”, The North River Press Publishing Corporation, Great Barrington.

Govoni, D. (1999) Java Application Frameworks, Wiley & Sons, ISBN 0-471-32930-4.

Graham, M., Scarborough, H. & Goodwin, C. (1999) Implementing Computer Mediated Communication in an Undergraduate Course - A Practical Experience. Journal of Asynchronous Learning Networks, Vol. 3, No. 1, 1999, pp. 32-45.

Greenberg, S. & Fitchett, C. (2001) Phidgets: Easy Development of Physical Interfaces through Physical Widgets. Proceedings of the UIST 2001 - 14th Annual ACM Symposium on User Interface Software and Technology, November 11-14, Orlando, Florida, ACM Press, pp. 209-218.

Greenberg, S. (2003) Enhancing Creativity with Groupware Toolkits. Invited keynote talk. Proceedings of the 9th International Workshop on Groupware (CRIWG 2003), Sept 28 - Oct 2, Autrans, France, LNCS vol. 2806, Springer-Verlag, pp. 1-9.

Greenberg, S. (2006) “Toolkits and Interface Creativity”, Journal of Multimedia Tools and Applications, Special Issue on Groupware, Kluwer. In Press. Disponível em http://grouplab.cpsc.ucalgary.ca/papers (consulta 15 de janeiro de 2006)

Greif, I. (1988) Computer Supported Cooperative Work - A book of readings. Morgan Kaufmann Publishers, USA, 1988. ISBN 0-934613-57-5.

Griss, M.L. (2001) Product-Line Architectures, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Gross, T. (1997) “Towards flexible support for cooperation: group awareness in shared workspaces,” DEXA’97, França, IEEE, Los Alamitos, CA, pp. 406-411.

Grosz, B.J. (1996) Collaborative systems, AI Magazine 17 (2), pp. 67–85

Groupware Patterns Swiki (2005) http://www.groupware-patterns.org

Grudin, J. (1989) Why Groupware Applications Fail: Problems In Design And Evaluation. Office: Technology and People, Vol. 4, No. 3, pp. 245-264.

Grundy, J.C., Mugridge, W.B. & Hosking, J.G. (1997) “A Java-based Componentware Toolkit for constructing Multi-view Editing Systems”, Proceedings of the 2nd Component Users' Conference (CUC'97), Munich July 15-18.

Guerrero, L. & Fuller, D. (1999) “A Web-Based OO Platform for the Development of Multimedia Collaborative Applications”, Decision Support Systems, v.27, n.3, pp.255-268.

Guicking, A., Tandler, P. & Avgeriou, P. (2005) “Agilo: A Highly Flexible Groupware Framework”, Proceeding of the 11th International Workshop on Groupware, CRIWG 2005, Porto de Galinhas-PE, September 2005, Fuks, H., Lukosch, S. & Salgado, A.C. (eds), Lecture Notes in Computer Science, Vol 3706, pp. 49-56.

Gutwin, C. & Greenberg, S. (2000) The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces. IEEE 9th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises -WETICE (2000), p. 98-103.

Gutwin, C. & Greenberg, S. (2002) A Descriptive Framework of Workspace Awareness for Real-Time Groupware, Computer Supported Cooperative Work, Vol. 11, No. 3-4, pp. 411-446.

Gutwin, C., Stark, G. & Greenberg, S. (1995) "Support for workspace awareness in educational groupware". Computer Support for Collaborative Learning, Lawrence Erlbaum Associates, New York, 1995, pp. 147-156.

Hansen, R.P., Pinto, S.C.C.S. & Hansen, C.R. (2005) “Integrando Web Services e Recursos Educacionais Através de Composição” XATA2005 - XML: Aplicações e Tecnologias Associadas, 2005, Braga, Portugal, pp. 290-301.

Harasim, L., Hiltz, S. R., Teles, L., & Turoff, M. (1997) “Learning networks: A field guide to teaching and online learning”, 3rd ed., MIT Press, 1997.

Heineman, G.T. (2000) “A model for designing adaptable software components”, ACM SIGSOFT Software Engineering Notes archive, Volume 25, Issue 1, January 2000, ISSN 0163-5948, pg 55-56

Hess, H., Cohen, S., Holibaugh, B., Kyang, K., Peterson, A.S., Novak, W. & Carroll, P. (2000) A Domain Analysis Bibliography, Carnegie Mellon University/Software Engineering Institute, Special Report 90-SR-3.

Houaiss (2001) Dicionário Eletrônico da Língua Portuguesa, versão 1.0, Editora Objetiva.

Houston, K. & Norris, D. (2001) Software Componentes and the UML, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Hummes, J. & Merialdo, B. (2000) Design of Extensible Component-Based Groupware. Computer Supported Cooperative Work, 9(1), 53-74. ISSN 0925-9724.

Hwang, M.I. & Lin, J.W. (1999) Information dimension, information overload and decision quality. Journal of Information Science. Vol. 25, no. 3, pp. 213-218.

Ierusalimschy, R., Figueiredo, L.H. & Celes, W (1996) “Lua – an extensible extension language. Software: Practice & Experience”, 26(6), 635-652.

Isenhour, P.L., Begole, J., Heagy, W.S. & Shaf-fer, C.A. (1997) “Sieve: A java-based collaborative visualization environment”, Late Breaking Hot Topics Proceedings, IEEE Visualization 97, 1997

Jacobson, I., Griss, M. & Jonsson, P. (1997) Software Reuse: Architecture, Process and Organization for Business Success, Addison-Wesley, Nova York.

Johnson, R. (1997) “Components, Frameworks, Patterns”, Symposium of Software Reusability 1997 USA.

Johnson, R. (2002): Expert One-on-One J2EE Design and Development. Reading: Wiley Publishing Inc., 2002.

Johnson, R. (2004): Expert One-on-One J2EE Development without EJB. Wiley Publishing Inc., 2004.

Kang, K., Cohen, S., Hess, J., Novak, W. & Peterson, A. (1990) “Feature-Oriented Domain Analysis (FODA) Feasibility Study” (CMU/SEI-90-TR-21). Pittsburg, PA, Software Engineering Institute, Carnegie Mellon University.

Kang, K., Kim, S., Lee, J., Kim, K., Shin E. & Huh, M. (1998) “FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering, 5, pp. 143-168

Kanselaar, G., Erkens, G., Andriessen, J., Prangsma, M., Veerman, A. & Jaspers, J. (2003) Designing Argumentation Tools for Collaborative Learning, Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making, Kirschner, P., Shum, S. & Carr, C. (eds.), chap. 3, Springer-Verlag.

Kirschner, P.A., Shum, S.J.B. & Carr, C.S. (2003) Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making, Springer.

Kirsch-Pinheiro, M., Lima, J.V. & Borges, M.R.S. (2002) A Framework for Awareness Support in Groupware Systems. Proceedings of the 7th International Conference on Computer Supported Cooperative Work in Design – CSCWD’2002, Rio de Janeiro, Brazil.

Koch, M. & Koch, J. (2000) “Application of Frameworks in Groupware – The Iris Group Editor Environment”, ACM Computing Surveys Symposium on Frameworks.

Kolfschoten, G.L., Briggs, R.O., Appelman, J.H. & de Vreede, G.J (2004) “ThinkLets as Building Blocks for Collaboration Processes: A Further Conceptualization Groupware”, Proceeding of the 10th International Workshop on Groupware, CRIWG 2004, San Carlos, Costa Rica, September 5-9, Lecture Notes in Computer Science, Vol 3198, pp. 137-152.

Kraut, R.E. & Attewell, P. (1997) “Media use in global corporation: electronic mail and organizational knowledge,” Research milestone on the information highway, Mahwah, NJ: Erlbaum.

Krebs, A.M., Ionescu, M., Dorohomceanu, B. & Marsic, I. (2003) “The DISCIPLE System for Collaboration over the Heterogeneous Web” Proceedings of the Hawaii International Conference on System Sciences – HICSS 2003.

Kreifelts, T., Hinrichs, E. & Woetzel, G. (1993) Sharing To-Do Lists with a Distributed Task Manager. In Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW’93), pp. 31-46, 1993.

Krueger, C.W. (1992) Software Reuse, ACM Computing Surveys, Volume 4 Issue 2 p131-183.

Kulesza, U., Garcia, A., Lucena, C.J.P. & Staa, A.V. (2004) “Integrating Generative and Aspect-Oriented Technologies”, Simpósio Brasileiro de Engenharia de Software 2004, Brasília, pp. 130-146.

Kuutti, K. (1991) “The concept of activity as a basic unit of analysis for CSCW research”, Proceedings of the Second European Conference on Computer Supported Cooperative Work (ECSCW’91), pp. 249-264.

Lajoie, R. & Keller, R.K. (1995) “Design and reuse in object-oriented frameworks: Patterns, contracts, and motifs in concert”, Object-Oriented Technology for Database and Software Systems, Alagar, V. & Missaoui, R. (eds), Singapore, 1995, World Scientific, pp. 295-312.

Lange, B.M. & Gershman, A. (1992) “OMNI: A Corporate Knowledge Environment for Collaborative Work”, IEEE Conference on Systems, Man and Cybernetics, 1992.

Larman, C. (2004) Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado, 2ª edição, Bookman, Porto Alegre.

Laufer, C. & Fuks, H. (1995) “ACCORD: Conversation Clichés for Cooperation”, Proceedings of The International Workshop on the Design of Cooperative Systems, Juan-les-Pins, França, pp 351-369.

Laurillau, Y. & Nigay, L. (2002) “Clover architecture for groupware”, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW 2002), pp. 236 - 245

Lee, D., Lim, M. & Han, S. (2002) “ATLAS: a scalable network framework for distributed virtual environments”, Proceedings of the 4th International Conference on Collaborative Virtual Environments.

Li, D. & Muntz, R.R. (1998) “COCA: Collaborative Objects Coordination Architecture”, Proceedings of the ACM Conference on Computer Supported Cooperative Work 1998, pp. 179-188.

Litiu, R. & Prakash, A. (2000) “Developing Adaptive Groupware Applications Using a Mobile Computing Framework”, Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'00), pp. 107-116.

Liu, Y., Shi, Y. & Xu, G. (2001) “Supporting Group Awareness in Collaborative Design”, Proceeding of Computer Supported Cooperative Work in Design 2001, London, Jul 2001.

Long, B. & Baecker, R. (1997) “A taxonomy of Internet communication tools”, Proceedings of WebNet - World Conference of the WWW, Internet, and Intranet, Toronto, Canada, ISBN 1-880094-27-4, pp. 318-323.

Lucena, C.J.P. & Fuks, H. (2000) Professores e Aprendizes na Web: A Educação na Era da Internet, ISBN 85-88011-01-8, Editora Clube do Futuro, Rio de Janeiro, 2000.

Lucena, C.J.P., Fuks, H., Milidiu, R., Macedo, L., Santos, N., Laufer, C., Blois, M., Fontoura, M.F., Choren, R., Pinto, S.C.C.S., Torres, V., Daflon, L. & Lukowiecki, L. (1998) AulaNet - An Environment for the Development and Maintenance of Courses on the Web. Proceedings of ICEE'98 - International Conference On Engineering Education, Rio de Janeiro, 1998

Lukosch, S. & Schümmer, T. (2004) Patterns for Managing Shared Objects in Groupware Systems, Proceedings of the 9th European Conference on Pattern Languages and Programs, Irsee, Germany.

Mackenzie, J. (1985) No Logic before Friday, Synthese, V.63, pp 329-341.

Magnusson, M. & Svensson, L. (2000) Studying how students study: Work-orientation and collaboration in Distance Education, Proceedings of IRIS 23, Svensson et al (eds.), University of Trollhättan Uddevalla. Sweden

Malone, T.W. & Crowston, K. (1990) What is Coordination Theory and How Can It Help Design Cooperative Work Systems? Conference on Computer-Supported Cooperative Work (CSCW), pp. 357-370.

Mandviwalla, M. & Olfman, L. (1994) What do groups need? A proposed set of generic requirements. ACM Transactions on Computer-Human Interaction, Vol. 1, No. 3, pp. 245-268.

Markus, M.L. & Connolly, T. (1990) Why CSCW applications fail: Problems in the adoption of interdependent work tools. Proceedings of CSCW'90 (Los Angeles, Oct. 7-10).

Marsic, I. & Dorohonceanu, B. (2003) “Flexible User Interfaces for Group Collaboration”. International Journal of Human-Computer Interaction, Vol.15, No.3, pp. 337-360

Marsic, I. (1999) DISCIPLE: a framework for multimodal collaboration in heterogeneous environments. ACM Computing Surveys, 31 (2es), Article No. 4.

Mattsson, M. (2000) Evolution and Composition of Object-Oriented Frameworks, PhD Thesis, Department of Software Engineering and Computer Science, University of Karlskrona/Ronneby.

McIlroy, M.D. (1968) "Mass Produced Software Components", Software Engineering, NATO Science Committee, pp. 138-150.

Michailidis, A. & Rada, R. (1996) “A review of collaborative authoring tools”, Groupware and authoring. Academic Press, London, 1996, p. 9-43.

Mitchell, L.H.R.G., Fuks, H. & Lucena, C.J.P. (2004) Contribuições da Gestão de Competências para a Educação a Distância: Experimento com o Ambiente AulaNet. Informática na Educação: Teoria e Prática, Vol 7, No. 2, Porto Alegre, UFRGS, ISSN 1516-084X, pp. 83-98.

Moore, J. M. & Bailin, S.C. (1991) “Domain Analysis: Framework For Reuse”, Prieto-Diaz, R. & Arango, G. (eds.), Domain Analysis and Software System Modeling. LosAlamitos, CA: IEEE Computer Society Press, pp. 179-203.

Morch, A.I. (1997) “Three Levels of End-user Tailoring: Customization, Integration, and Extension” In: Computers and Design in Context, Edited by M. Kyng and L. Mathiassen, MIT Press, USA.

Motta, C.L.R. & Borges, M.R.S. (2000) “A Cooperative Approach for Information Recommendation and Filtering”, Proceedings of the International Workshop on Groupware, IEEE Computer Society, Madeira, Portugal, October 2000, pp. 42-49.

Muhammad, A., Enríquez, A.M.M. & Decouchant, D. (2005) “Awareness and Coordination for Web Cooperative Authoring”, Advances in Web Intelligence: Third International Atlantic Web Intelligence Conference, AWIC 2005, Lodz, Poland, June 6-9, 2005. ISBN: 3-540-26219-9.

Myers, B. (1995) “State of the Art in User Interface Software Tools”, In Baecker, R., Grudin, J., Buxton, W. & Greenberg, S. (eds), *Readings in Human Computer Interaction: Towards the Year 2000*. Morgan Kaufmann , pp. 323-343.

Neale, D.C., Carroll, J.M. & Rosson, M.B. (2004) “Evaluating computer-supported cooperative work: models and frameworks”, Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04), Chicago, Illinois, USA, November 06 - 10, ACM Press, New York, pp. 112-121.

Neisser, U. (1976) Cognition and Reality, Ed. W.H. Freeman, San Francisco.

Nielsen, J. (1994) “Heuristic Evaluation”, Usability Inspection Methods, Chapter 2, Nielsen, J. & Mack, R. (eds), John Wiley and Sons, New York, pp. 25-62.

Nunamaker, J.F., Romano, N.C. & Briggs, R.O. (2001) A Framework for Collaboration and Knowledge Management. In: Proceedings of 34th Hawaii International Conference on System Sciences - HICSS'01.

Oliveira, R.F., Blois, A.P.T.B., Vasconcelos, A.P.V. & Werner, C.M.L. (2005) “Representação de Variabilidades em Componentes de Negócio no Contexto da Engenharia de Domínio”, V Workshop em Desenvolvimento Baseado em Componentes, Juiz de Fora, MG, novembro 2005, pp. 73-80.

Oliveira, T.C. (2001) Uma Abordagem Sistemática para a Instanciação de Frameworks Orientados a Objetos, Tese de Doutorado, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, 2001.

OMG (2005) Unified Modeling Language Superstructure Specification, version 2.0, formal/05-07-04, Agosto de 2005.

Orfali, R., Harkey, D. & Edwards, J. (1996) The Essential Distributed Objects Survival Guide, John Wiley & Sons, New York.

Ostwald, J. (1995) "Supporting collaborative design with representations for mutual understanding", CHI'95 Proceedings, Doctoral Consortium.

Osuna, C.A. & Dimitriadis, Y.A. (1999) "A Framework for Development of Educacional-Collaborative Applications Based on Social Constructivism". Proceedings of International Workshop on Groupware - CRIWG'99, IEEE Press, Cancun, Mexico.

Paludo, M.A. & Burnett, R.C. (2005) "Desenvolvimento Baseado em Componentes e Padrões", in: Desenvolvimento Baseado em Componentes, Gimenes, I.M.S. & Huzita, E.H.M. (eds), Editora Ciência Moderna, Rio de Janeiro, 2005. ISBN 85-7393-406-9, pg. 199-231.

Peterson, A.S. (1991) Coming to Terms with Software Reuse Terminology: a Model-based Approach, ACM SIGSOFT Software Engineering Notes, Volume 16, Issue 2, April 1991, pp. 45-51, ISSN:0163-5948

Pfleeger, S.L. (2001) Software engineering: theory and practice, Upper Saddle River, NJ: Prentice Hall.

Philippe, K. (2003): Introdução ao RUP – Rational Unified Process. Rio de Janeiro: Ciência Moderna.

Pimentel, M. (2006) RUP-3C-Groupware: um processo de desenvolvimento de groupware baseado no Modelo 3C de Colaboração, Tese de Doutorado, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, 2006.

Pimentel, M., Fuks, H. & Lucena, C.J.P. (2003a) Debati, debati... aprendi? Investigações sobre o papel educacional das ferramentas de bate-papo. WIE 2003 - IX Workshop de Informática na Escola, Anais do XXIII Congresso da Sociedade Brasileira de Computação, V5, Campinas-SP, 2 a 8 de agosto de 2003. pp. 167-178.

Pimentel, M., Fuks, H. & Lucena, C.J.P. (2003b) Co-text Loss in Textual Chat Tools. 4th International and Interdisciplinary Conference on Modeling and Using Context - CONTEXT 2003, LNAI 2680, Stanford, CA, USA, June, pp 483-490, 2003.

Pimentel, M., Fuks, H. & Lucena, C.J.P. (2004) "Avaliação da Participação em Conferências Textuais Assíncronas", Anais Eletrônico do X Workshop de Informática na Escola, integrante do XXIV Congresso da Sociedade Brasileira de Computação (WIE/SBC 2004), ISBN: 85-88442-94-9, 31 Julho - 6 Agosto, Salvador, BA.

Pimentel, M., Fuks, H. & Lucena, C.J.P. (2005) "Mediated Chat Development Process: Avoiding Chat Confusion on Educational Debates", Computer Supported Collaborative Learning, Taiwan, July 2005, ISBN 0805857826, Lawrence Erlbaum Associates, pp. 499-503.

Pinto, S.C.C.S. (2000) Composição em WebFrameworks, tese de doutorado, Departamento de Informática PUC-Rio.

POJO (2005): Trecho do blog do Martin Fowler onde o termo Plain Old Java Object é explicado disponível em http://www.martinfowler.com/bliki/POJO.html Última visita em 20/10/2005.

Prakash, A. & Knister, M.J. (1994) "A framework for undoing actions in collaborative systems", ACM Transactions on Computer-Human Interaction, 1(4)295–330, December 1994.

Pree, W. (1995) Design Patterns for Object-Oriented Software Development, Addison-Wesley Publishing Company, 1995.

Pressman, R.S. (2000) Software Engineering : A Practitioner's Approach, McGraw Hill, New York, NY, June 2000.

Pumareja, D., Sikkel, K. & Wieringa, R. (2004) “Understanding the dynamics of requirements evolution: a comparative case study of groupware implementation”, Proceedings of the 10th Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2004), Essener Informatik Beiträge 9, pp. 177-194.

Raposo, A.B. & Fuks, H. (2002) Defining Task Interdependencies and Coordination Mechanisms For Collaborative Systems. In M. Blay-Fornarino, A.M. Pinna-Dery, K. Schmidt and P. Zaraté (eds) Cooperative Systems Design (Frontiers In Artificial Intelligence and Applications Vol. 74). IOS Press, Amsterdam, pp. 88-103.

Raposo, A.B., Gerosa, M.A. & Fuks, H. (2004) “Combining Communication and Coordination toward Articulation of Collaborative Activities”, 10th International Workshop on Groupware - CRIWG 2004. Vreede, G.J., Guerrero,L.A., Raventós,G.M. (eds.). Lecture Notes on Computer Science LNCS Vol. 3198, ISBN 3540-230165, ISSN 0302-9743. San Carlos, Costa Rica: Springer-Verlag, 5-9 September 2004. pp. 121-136.

Rezende, J.L., Fuks, H. & Lucena, C.J.P. (2003) Aplicando o Protocolo Social através de Mecanismos de Coordenação embutidos em uma Ferramenta de Bate-Papo. XIV Simpósio Brasileiro de Informática na Educação - SBIE 2003, 12 a 14 de Novembro de 2003, ISBN: 85-88442-70-1, Rio de Janeiro - RJ, pp. 55-64.

Romano, D., Brna, P. & Self, J. (1998) “Collaborative decision-making and presence in shared dynamic virtual environments”, Proceedings of the Workshop on Presence in Shared Virtual Environments. BT Labs, Martlesham Heath.

Roschelle, J. & Teasley, S. (1995) The construction of shared knowledge in collaborative problem solving. In O’Malley, C.E., (ed.), Computer Supported Collaborative learning. Springer-Verlag, Heildelberg, pp 69-97.

Roseman, M. & Greenberg, S. (1996) “Building real time groupware with GroupKit, a groupware toolkit”. ACM Transactions on Computer-Human Interaction, 3, March 1996, 1, p. 66-106.

Roth, J. & Unger, C. (2000) Developing synchronous collaborative applications with TeamComponents. In Designing Cooperative Systems: the Use of Theories and Models, Proceedings of the 5th International Conference on the Design of Cooperative Systems (COOP’00), pp. 353-368.

Roussos, M., Johnson, A.E., Leigh, J., Bames, C.R., Vasilakis, C.A. & Moher, T.G. (1997) “The NICE Project: Narrative, Immersive, Constructionist/Collaborative Environments for Learning in Virtual Reality”. Proceedings of Educational Multimedia and Telecommunications- ED-Media, 1997.

Rubart, J. & Dawabi, P. (2002) Towards UML-G: A UML-Profile for Modeling Groupware, 8th International Workshop on Groupware, CRIWG 2002, LNCS 2440 , Springer-Verlag, La Serena, Chile, 2002, pp. 93-113.

Sacramento, V., Endler, M., Rubinsztejn, H.K., Lima, L.S., Gonçalves, K., Nascimento, F.N. & Bueno, G.A. (2004) "MoCA: A Middleware for Developing Collaborative Applications for Mobile Users," IEEE Distributed Systems Online, vol. 5, no. 10, 2004. ISSN 1541-4922

Salmon, G. (2000) E-moderating: the key to teaching and learning online, London, Kogan Page

Salus, P.H. (1998) "Handbook of Programming Languages: Object-Oriented Programming Languages". Vol. 1. Macimillian. Indianapolis. 1998.

Sametinger, J. (1997) Software Engineering with Reusable Components. Springer Verlag, USA, 1997.

Santoro, F.M., Borges, M.R. & Santos, N. (2000) "An Infrastructure to Support the Development of Collaborative Project-Based Learning Environments" Proceedings of International Workshop on Groupware – CRIWG00, Madeira, Portugal, IEEE Press, pp. 78-85.

Santoro, F.M., Borges, M.R. & Santos, N. (2001) "Modelo de Cooperação para Aprendizagem Baseada em Projetos: Uma Linguagem de Padrões". In: 1ª Conferência Latino Americana em Linguagens de Padrão para Programação – SugarLoafPloP. Rio de Janeiro, Outubro de 2001.

Sauter, C., Morger, O., Muhlherr, M., Thutchytson, A. & Teusel, S. (1995) CSCW for Strategic Management in Swiss Enterprises: an Empirical Study. Proceedings of the 4th European Conference on Computer Supported Cooperative Work (ECSCW'95), Stockholm, Sweden, 117-132

Schmidt, K. & Rodden, T. (1996) Putting it all Together: Requirements for a CSCW Platform. In: Shapiro, D., Tauber, M., Traunmüller, R. (eds.) The Design of Computer Supported Cooperative Work and Groupware Systems. North Holland, Holland, pp. 157-176.

Schmidt, K. & Simone, C. (1996) Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. Computer Supported Cooperative Work, 5(2-3), 155-200.

Schmidt, K. (1991) "Riding a Tiger, or Computer Supported Cooperative Work", Proceedings of The 2nd European Conference on Computer-Supported Cooperative Work, Kluwer, 1-16.

Schön, D. & Bennet, J. (1996) "Reflective Conversation with Materials", In: Bringing Design to Software, Winograd, T. (ed), ACM Press, USA. ISBN 0-201-85491-0

Schön, D.A. (1983) The reflective practitioner: How professionals think in action, Basic Books, NY

Schrage, M. (1995) No more teams! Mastering the dynamics of creative collaboration, Nova York, EUA: Currency Doubleday

Schrage, M. (1996) Cultures Of Prototyping. In T. Winograd (ed) Bringing Design To Software, ACM Press, USA, pp. 191-205.

Schwarz, J., Sommer, H. & Farris, A. (2003) The ALMA Software System, ASP Conf. Ser., Vol. 314 Astronomical Data Analysis Software and Systems XIII, Ochsenbein, F., Allen, M. & Egret, D. (eds), San Francisco, ASP, 643

Segal, L. (1994) "Effects of Checklist Interface on Non-Verbal Crew Communications," Contractor Report 177639, NASA Ames Research Center.

Shum, S.B. & Hammond, N. (1994) “Argumentation-based design rationale: what use at what cost?,” Human-Computer Studies, USA, 40, p. 603-652.

Siebra, S.A., Salgado, A.C., Tedesco, P.A. & Brézillon, P. (2005) “Identifying the Interaction Context in CSCLE”, Proceedings of the 5th International and Interdisciplinary Conference (CONTEXT 2005), Paris, France, July 5-8, Lecture Notes in Computer Science, Vol 3554, Springer-Verlag.

Siegel, J. (2000) CORBA 3 Fundamentals and Programming, 2nd Edition, John Wiley & Sons, ISBN 0471295183

Simon, H.A. (1996) Models of my life, MIT Press, ISBN 0-262-69185-X

Simon, J.A. (1997) “Towards Mixed-Initiative Discursive Networking”, Computational Models for Mixed Initiative Interaction, March 24-26, 1997, Stanford University, California.

Siqueira, S.W.M., Braz, M.H.L.B. & Melo, R.N. (2003) “E-Learning Environment Based on Framework Compositio”, Third IEEE International Conference on Advanced Learning Technologies (ICALT'03), pp. 468.

Sire, S., Chatty, S., Gaspard-Boulinc, H. & Colin, F. (1999) How can groupware preserve our coordination skills? Designing for direct collaboration. Proceedings of Human-Computer Interaction INTERACT’99, IOSPress, pp 304–312.

Slagter, R.J. & Biemans, M.C.M. (2000) “Component Groupware: A Basis for Tailorable Solutions that Can Evolve with the Supported Task”, in Proceedings of the International ICSC Conference on Intelligent Systems and Applications (ISA 2000), Wollongong, Australia.

Slagter, R.J. & ter Hofte, H. (1999) Component models and component groupware architectures. TI/RS/99023, GIGACSCW/D2.1.1, Telematica Instituut, the Netherlands.

Slagter, R.J., Biemans, M, & Ter Hofte, G.H. (2001) “Evolution in use of groupware: Facilitating tailoring to the extreme” In: M. Borges, J. Haake and U. Hoppe (eds.), Proceedings of the 7th International Workshop on Groupware (CRIWG 2001), 6-8 September 2001, Darmstadt, Germany, 2001

Sommerville, I. (2003): Engenharia de Software. 6 ed. São Paulo: Addison Wesley, 2003.

Stafford, J.A. & Wolf, A.L. (2001) Software Architecture, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Stahl, G. (2001) WebGuide: Guiding collaborative learning on the Web with perspectives, Journal of Interactive Media in Education.

Stiemerling, O., Hallenberger, M. & Cremers, A.B. (2001) "A 3D Interface for the Administration of Component-Bases, Distributed Systems", Proceedings of the 5th International Symposium on Autonomous Decentralized Systems, p. 119.

Stiemerling, O., Hinken, R. & Cremers, A.B. (1999) The EVOLVE Tailoring Platform: Supporting the Evolution of Component-Based Groupware. In Proceedings of the 3rd International Enterprise Distributed Object Computing Conference (EDOC’99), pp. 106-115.

Streitz, N., Haake, J., Hannemann, J., Lemke, A., Schuler, W., Scheutt, H. & Thuring M. (1992) SEPIA: a cooperative hypermedia authoring environment. Proceedings of Proceedings of ACM Conference on Hypertext (ECHT'92), pp. 11-22.

Szyperski, C. (1997) Component Software: Beyond Object-Oriented Programming, Addison-Wesley, ISBN 0-201-17888-5

Szyperski, C. (2003) Component technology – what, where, and how? Procedings of the 25th International Conference on Software Engineering (ICSE'03), IEEE Computer Society, pp 684-693.

Tam, J. & Greenberg, S. (2004) A framework for asynchronous change awareness in collaboratively-constructed documents, CRIWG 2004, LNCS 3198, p. 67-83.

Tapscoot, D., Ticoll, D. & Lowy, A. (2000) Digital Capital: Harnessing the Power of Business Webs. Harvard Business School Press, USA.

Tatikonda, M.V & Stock, G.N. (2003) "Product Technology Transfer in the Upstream Supply Chain", The Journal of Product Innovation Management, Vol. 20, Product Development & Management Association, pp. 444-467

Teege, G. (1996) Object-Oriented Activity Support: A Model for Integrated CSCW Systems. Computer Supported Cooperative Work (CSCW): The Journal of Collaborative Computing, 5(1), pp. 93-124, 1996.

Teixeira, B. & Chagas, E.F. (2005) "Co-Autoria: Avaliação e Proposta de Requisitos para Ferramentas Segundo o Modelo 3C", Workshop de Informática na Escola, Congresso da Sociedade Brasileira de Computação 2005.

Teles, V.M. (2004) Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade, ed. Novatec, ISBN 8575220470.

Tietze, D.A., A Framework for Developing Component-based Co-operative Applications. PhD Dissertation, Computer Science, Technischen Universität Darmstadt, Germany, 2001.

Tripathi, A., Ahmed, T., Kumar, R. & Jaman, S. (2002) "Design of a Policy-Driven Middleware for Secure Distributed Collaboration", Proceedings of the 22nd International Conference on Distributed Computing System (ICDCS), Vienna, Austria, July 2002.

Tse, E. & Greenberg, S. (2004) "Rapidly Prototyping Single Display Groupware through the SDGToolkit", Proceedings of the 5th Australasian User Interface Conference, Volume 28 in the CRPIT Conferences in Research and Practice in Information Technology Series, Dunedin, NZ, pp. 101-110.

van der Veer, G.C., Lenting, B.F. & Bergevoet, B.A.J. (1996) GTA: Groupware Task Analysis - Modeling Complexity. Acta Psychologica 91, 1996, pp. 297-322

Wan, D. & Johnson, P.M. (1994) "Computer Supported Collaborative Learning Using CLARE: the Approach and Experimental Findings". Proceedings of ACM Conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina, 1994.

Weinreich, R. & Sametinger, J. (2001) Component Models and Component Services: Concepts and Principles, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Werner, C.M.L. & Braga, R.M.M.B (2005) "A Engenharia de Domínio e o Desenvolvimento Baseado em Componentes", in: Desenvolvimento Baseado em Componentes, Gimenes, I.M.S. & Huzita, E.H.M. (eds), Editora Ciência Moderna, Rio de Janeiro, 2005. ISBN 85-7393-406-9, pg. 57-103.

Whorf, B.L. (1956) Language, Thought and Reality. Cambridge, MA: MIT Press.

Wills, A.L. (2001) Components and Connectors: Catalysis Techniques for Designing Component Infrastructures, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

Winograd, T. & Flores, F. (1987) Understanding Computers and Cognition. Addison-Wesley, USA, 1987.

Won, M., Stiemerling, O. & Wulf, V. (2005) "Component-Based Approaches to Tailorable Systems", End User Development, Lieberman, H.;, Paterno, F. & Wulf, V. (eds), Kluwer, pp. 1-27.

Yang, Y. (1995) "Coordination for process support is not enough!" European Workshop on Software Process Technology (EWSPT4), LNCS 913, Leiden, The Netherlands, April 1995. Springer Verlag.

Artigos Publicados pelo Autor da Tese

Os artigos estão disponíveis para download no site http://groupware.les.inf.puc-rio.br

Capítulos de Livros

Fuks H., Pimentel, M.G., **Gerosa, M.A.**, Fernandes, M.C.P. & Lucena, C.J.P. (2006), "Novas Estratégias de Avaliação Online: Aplicações e Implicações em um Curso totalmente a Distância através do Ambiente AulaNet", *EaD Online: Avaliação* (título provisório), Silva, M. (ed.), Edições Loyola, Rio de Janeiro. (a ser publicado)

Fuks H., **Gerosa, M.A.** & Lucena, C.J.P. (2003), "Using the AulaNet Learning Environment to Implement Collaborative Learning via Internet", in: *Innovations 2003 – World Innovations in Engineering Education and Research*, Aung et al. (ed), iNEER, USA, 2003, Chap. 23, ISBN 0-9741252-0-2, pp. 225-235.

Fuks H., Cunha, M.L., **Gerosa, M.A.** & Lucena, C.J.P. (2003), "Participação e Avaliação no Ambiente Virtual AulaNet da PUC-Rio", *EaD Online: Teorias e Práticas*, Silva, M. (ed.), Edições Loyola, Rio de Janeiro, 2003, ISBN 85-15-02822-0, Cap. 15, pp. 231-254.

Fuks, H., **Gerosa, M.A.** & Pimentel, M.G. (2003), "Projeto de Comunicação em Groupware: Desenvolvimento, Interface e Utilização", *XXII Jornada de Atualização em Informática*, Anais do XXIII Congresso da Sociedade Brasileira de Computação, V2, Cap. 7, ISBN 85-88442-59-0, pp. 295-338.

Fuks, H., Raposo, A.B. & **Gerosa, M.A.** (2002), "Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas", *XXI Jornada de Atualização em Informática*, Anais do XXII Congresso da Sociedade Brasileira de Computação, V2, Cap. 3, ISBN 85-88442-24-8, pp. 89-128.

Periódicos / Journals

Fuks, H., **Gerosa, M.A.**, Pimentel, M.G., Filippo, D. & Lucena, C.J.P. (2006), "Informações Estatísticas e Visuais para a Mediação de Fóruns Educacionais", *Revista Brasileira de Informática na Educação*, ISSN 1414-5685, Sociedade Brasileira de Computação. (aceito para publicação)

Fuks, H., Raposo, A.B., **Gerosa, M.A.**, Lucena, C.J.P. (2005), "Applying the 3C Model to Groupware Development", *International Journal of Cooperative Information Systems* (IJCIS), v.14, n.2-3, Jun-Sep 2005, World Scientific, ISSN 0218-8430, pp. 299-328.

Fuks, H., **Gerosa, M.A.**, Raposo, A.B. & Lucena, C.J.P. (2004), "O Modelo de Colaboração 3C no Ambiente AulaNet", *Informática na Educação: Teoria e Prática*, Vol 7, No. 1, Porto Alegre, UFRGS, ISSN 1516-084X, pp. 25-48.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2003), "Analysis and Design of Awareness Elements in Collaborative Digital Environments: A Case Study in the AulaNet Learning Environment", *The Journal of Interactive Learning Research*, V. 14, No. 3, AACE, USA, ISSN 1093-023X, pp. 315-332.

Gerosa, M.A., Fuks, H., & Lucena, C.J.P. (2003), "Suporte à Percepção em Ambientes de Aprendizagem Colaborativa", *Revista Brasileira de Informática na Educação*, V. 11, No. 2, Julho/Dezembro 2003, ISSN 1414-5685, Sociedade Brasileira de Computação, pp. 75-85.

Fuks, H., **Gerosa, M.A.** & Lucena, C.J.P. (2002), "The Development and Application of Distance Learning on the Internet", *Open Learning Journal*, V. 17, No. 1, February 2002, ISSN 0268-0513, Cartafax Pub, pp. 23-38.

Fuks, H., **Gerosa, M.A.** & Lucena, C.J.P. (2002), "Usando a Categorização e Estruturação de Mensagens Textuais em Cursos pelo Ambiente AulaNet", *Revista Brasileira de Informática na Educação*, V. 10, No. 1, Abril 2002, ISSN 1414-5685, Sociedade Brasileira de Computação, pp. 31-44.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001), "Tecnologias de Informação Aplicadas à Educação: Construindo uma Rede de Aprendizagem Usando o Ambiente AulaNet", *Informática na Educação: Teoria e Prática*, V. 4, No. 2, dezembro 2001, ISSN 1516-084X, UFRGS, pp. 63-74.

Fuks, H., **Gerosa, M.A.** & Lucena, C.J.P. (2001), "Sobre o Desenvolvimento e Aplicação de Cursos Totalmente a Distância na Internet", *Revista Brasileira de Informática na Educação*, No. 9, Setembro 2001, ISSN 1414-5685, Sociedade Brasileira de Computação, pp 61-75.

Anais de Congressos / Conferências

Gerosa, M.A., Pimentel, M.G., Filippo, D., Barreto, C.G., Raposo, A.B., Fuks, H. & Lucena, C.J.P. (2005) "Componentes Baseados no Modelo 3C para o Desenvolvimento de Ferramentas Colaborativas", *Anais do 5º Workshop de Desenvolvimento Baseado em Componentes - WDBC 2005*, 07-09 de Novembro, Juiz de Fora-MG, ISBN 85-88279-47-9, pp. 109-112.

Pimentel, M.G., **Gerosa, M.A.**, Filippo, D., Barreto, C.G., Raposo, A.B., Fuks, H. & Lucena, C.J.P. (2005) "AulaNet 3.0: desenvolvendo aplicações colaborativas baseadas em componentes 3C", *Anais do WCSCW2005 - Workshop Brasileiro de Tecnologias para Colaboração*, 07-08 de Novembro, Juiz de Fora-MG, pp. 761-770.

Gerosa, M.A., Pimentel, M.G., Fuks, H. & Lucena, C.J.P. (2005) "No Need to Read Messages Right Now: Helping Mediators to Steer Educational Forums Using Statistical and Visual Information", *Proceedings of the Computer Supported Collaborative Learning Conference – CSCL 2005*, 01-04 June, Taipei, Taiwan, ISBN 0-8058-5782-6, pp. 160-169.

Pimentel, M.G., **Gerosa, M.A.**, Fuks, H. & Lucena, C.J.P. (2005) “Assessment of Collaboration in Online Courses”, *Proceedings of the Computer Supported Collaborative Learning Conference – CSCL 2005*, 01-04 June, Taipei, Taiwan, ISBN 0-8058-5782-6, pp. 494-498.

Gerosa, M.A., Pimentel, M., Raposo, A.B., Fuks, H., Lucena, C.J.P., “Towards an Engineering Approach for Groupware Development: Learning from the AulaNet LMS Development”, *Proceedings of the 9^{th} International Conference on CSCW in Design (CSCWiD)*, Vol. 1, Coventry, U.K., May 2005, ISBN 1-84600-004-1, pp. 329-333.

Gerosa, M.A., Raposo, A.B., Fuks, H. & Lucena, C.J.P. (2004) “Uma Arquitetura para o Desenvolvimento de Ferramentas Colaborativas para o Ambiente de Aprendizagem AulaNet”, *Anais do XV Simpósio Brasileiro de Informática na Educação – SBIE 2004*, 09-12 de Novembro, Manaus-AM, ISBN 85-7401-161-4, pp. 168-177.

Fuks, H., Raposo, A.B., **Gerosa, M.A.**, Pimentel, M.G. & Lucena, C.J.P. (2004) “Suporte à Coordenação e à Cooperação em uma Ferramenta de Comunicação Textual Assíncrona: Um Estudo de Caso no Ambiente AulaNet”, *Anais do I Workshop Brasileiro de Tecnologias para Colaboração – WCSCW 2004*, em conjunto ao WebMidia & LA-Web 2004, 13-14 de Outubro, Ribeirão Preto-SP, ISBN 85-7669-010-1, pp. 173-180.

Gerosa, M.A., Barreto, C.G, Raposo, A.B., Fuks, H. & Lucena, C.J.P. (2004) “O Uso de uma Arquitetura Baseada em Componentes para Incrementar um Serviço do Ambiente AulaNet”, *Anais do 4º Workshop de Desenvolvimento Baseado em Componentes* - *WDBC 2004*, 15-17 de Setembro, João Pessoa-PB, ISBN 85-7669-001-2, pp. 55-60.

Raposo, A.B., **Gerosa, M.A.** & Fuks, H. (2004), "Combining Communication and Coordination toward Articulation of Collaborative Activities", *10^{th} International Workshop on Groupware - CRIWG 2004*, 5-9 September, San Carlos, Costa Rica, Lecture Notes on Computer Science LNCS 3198, Springer-Verlag, ISBN 3540-230165, ISSN 0302-9743, pp. 121-136.

Gerosa, M.A., Pimentel, M.G., Fuks, H. & Lucena, C.J.P. (2004), "Analyzing Discourse Structure to Coordinate Educational Forums", *The 7^{th} International Conference on Intelligent Tutoring Systems – ITS 2004*, Maceió-AL, August 30th to September, 3rd 2004, Lecture Notes on Computer Science LNCS 3220, Springer-Verlag, ISBN 3540-229485, ISSN 0302-9743, pp. 262-272.

Raposo, A.B., Pimentel, M.G., **Gerosa, M.A.**, Fuks, H. & Lucena, C.J.P. (2004), "Prescribing e-Learning Activities Using Workflow Technologies", *The 1^{st} International Workshop on Computer Supported Activity Coordination – CSAC 2004*, International Conference on Enterprise Information Systems – ICEIS 2004, Porto, Portugal, April 13, 2004, pp. 71-80. ISBN 972-8865-08-2

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2004), "Estruturação e Categorização de Mensagens em Ferramentas de Comunicação Textuais Assíncronas", Electronic *Proceedings of the World Congress on Engineering and Technology Education* - *WCETE'2004*, March 14-17, Santos-SP.

Gerosa, M.A., Fuks, H., Raposo, A.B. & Lucena, C.J.P. (2004), "Awareness Support in the AulaNet Learning Environment", *Proceedings of the IASTED International Conference on Web-Based Education* - *WBE 2004*, ACTA Press, February 16-18, Innsbruck, Austria, pp. 490-495.

Gerosa, M.A., Raposo, A.B., Fuks, H., Lucena, C.J.P. (2003), "Combinando Comunicação e Coordenação em Groupware", *3ª Jornada Ibero-Americana de Engenharia de Software e Engenharia de Conhecimento – JIISIC 2003*, 26-28 de Novembro, Valdivia, Chile, pp. 145-154.

Gerosa, M.A., Pimentel, M.G., Fuks, H. & Lucena, C.J.P. (2003), "Coordenação de Fóruns Educacionais: Encadeamento e Categorização de Mensagens", *XIV Simpósio Brasileiro de Informática na Educação – SBIE 2003*, 12-14 de Novembro, Rio de Janeiro-RJ, pp. 45-54.

Fuks, H., Raposo, A.B. & **Gerosa, M.A.** (2003) "Do Modelo de Colaboração 3C à Engenharia de Groupware", *IX Simpósio Brasileiro de Sistemas Multimídia e Web – Webmidia 2003*, Trilha especial de Trabalho Cooperativo Assistido por Computador, 03-06 de Novembro, Salvador-BA, pp. 445-452.

Fuks, H., Mitchell, L.H.R.G, **Gerosa, M.A.** & Lucena, C.J.P. (2003) "Competency Management for Group Formation in the AulaNet Learning Environment", *9^{th} International Workshop on Groupware - CRIWG 2003*, 28 September - 02 October, Grenoble, France, ISBN: 3-540-20117-3, Lecture Notes in Computer Science 2806, Springer-Verlag, pp. 183-190.

Raposo, A.B., **Gerosa, M.A.** & Fuks, H. (2003) "Modeling Coordination in Business-Webs", *Proceedings of the 3^{rd} IFIP Conference on E-commerce, E-business and E-government*, 21-24 September 2003, Guarujá-SP, pp. 549-559.

Fuks, H., **Gerosa, M.A.**, Pimentel, M.G., Raposo, A.B., Mitchell, L.H.R.G. & Lucena, C.J.P. (2003) "Evoluindo para uma Arquitetura de Groupware Baseada em Componentes: o Estudo de Caso do Learningware AulaNet", *Anais eletrônico do 3º Workshop de Desenvolvimento Baseado em Componentes - WDBC 2003*, 10-12 de Setembro, São Carlos-SP.

Mitchell, L.H.R.G., **Gerosa, M.A.** & Fuks, H. (2003) "Comparação da Resolução Colaborativa de Problemas em Sala de Aula e através do Ambiente AulaNet", *WIE 2003 - IX Workshop de Informática na Escola*, Anais do XXIII Congresso da Sociedade Brasileira de Computação, V5, 2-8 de agosto. Campinas-SP, pp. 135-147.

Fuks, H., Cunha, L.M., **Gerosa, M.A.** & Lucena, C.J.P. (2003), "Analyzing and Assessing Collaborative Learning Activities in a Web-Based Environment", *Electronic Proceedings of the ICEE 2003 – International Conference on Engineering Education*, July 21-25, Valencia, Spain.

Fuks, H., Raposo, A.B. & **Gerosa, M.A.** (2002), "Engineering Groupware for E-Business", *First Seminar on Advanced Research in Electronic Business (EBR'2002)*, November 7-8, Rio de Janeiro-RJ, pp. 78-84.

Gerosa, M.A., Fuks, H., Raposo, A.B. & Mitchell, L.H.R.G. (2002), "Using Groupware Tools to Extend the Organizational Memory with Collaboration Aspects", *The 7^{th} International Conference on CSCW in Design (CSCWiD)*, September 25-27, Rio de Janeiro-RJ, pp. 314-319.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2002), "Resultados da avaliação de um curso baseado na Web", *VIII Workshop de Informática na Escola - WIE 2002*, XXII Congresso da Sociedade Brasileira de Computação, V. 5, 17-19 de julho, Florianópolis-SC, pp. 477-485.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2002), "AulaNet-eLabora: Developing a groupware to bring the work and learning environment together", *Electronic Proceedings of the 7th International Conference on Engineering and Technology Education - INTERTECH 2002*, March 17-20, Santos-SP.

Fuks, H., **Gerosa, M.A.** & Lucena, C.J.P. (2002), "Using a Groupware Technology to Implement Cooperative Learning via the Internet - A case study", *Electronic Proceedings of the 35th Annual Hawaii International Conference on System Sciences – HICSS 35*, January 7-10, Hawaii.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001), "Elementos de percepção como forma de facilitar a colaboração em cursos via Internet", *XII Simpósio Brasileiro de Informática na Educação – SBIE 2001*, 21-23 de Novembro, Vitória-ES, pp. 194-202.

Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001), "Use of categorization and structuring of messages in order to organize the discussion and reduce information overload in asynchronous textual communication tools", *7th International Workshop on Groupware - CRIWG 2001*, September 6-8, Germany, pp 136-141.

Fuks, H., **Gerosa, M.A.**, Cunha, L.M. & Lucena, C.J.P. (2001), "Groupware Technology for cooperative learning via the Internet", *Electronic Proceedings of the International Conference on Engineering Education - ICEE 2001*, August 6-10, Oslo, Norway.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2001), "The application and enhancement of web based courses", *IASTED International Conference on Computers and Advanced Technology in Education – CATE 2001*, June 27-29, Banff, Canada, pp 8-11.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2001), "Uso da categorização e estruturação de mensagens para dinamizar a discussão e reduzir a sobrecarga de informação em cursos via Internet", *VII Workshop de Informática na Escola – WIE 2001*, Anais Eletrônicos do XXI Congresso da Sociedade Brasileira de Computação, 31 de julho a 2 de Agosto, Fortaleza-CE.

Cunha, L.M., **Gerosa, M.A.**, Fuks, H. & Lucena, C.J.P. (2001), "Desenvolvimento e aplicação de cursos totalmente a distância na Internet", *VII Workshop de Informática na Escola – WIE 2001*, Anais Eletrônicos do XXI Congresso da Sociedade Brasileira de Computação, 31 de julho a 2 de Agosto, Fortaleza-CE.

Gerosa, M.A., Cunha, L.M., Fuks, H. & Lucena, C.J.P. (2001), "Um groupware baseado no ambiente AulaNet desenvolvido com componentes", *Anais eletrônicos do 1º Workshop de Desenvolvimento Baseado em Componentes*, 21-22 Junho, Maringá-PR.